Post-silicon Validation of Radiation Hardened Microprocessor, Embedded Flash and Test Structures

by

Anudeep Reddy Gogulamudi

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2016 by the
Graduate Supervisory Committee:

Lawrence T. Clark, Chair
Keith E. Holbert
John Brunhaver

ARIZONA STATE UNIVERSITY

May 2016

ABSTRACT

Digital systems are essential to the technological advancements in space exploration. Microprocessor and flash memory are the essential parts of such a digital system. Space exploration requires a special class of radiation hardened microprocessors and flash memories, which are not functionally disrupted in the presence of radiation. The reference design ‘HERMES’ is a radiation-hardened microprocessor with performance comparable to commercially available designs. The reference design ‘eFlash’ is a prototype of soft-error hardened flash memory for configuring Xilinx FPGAs. These designs are manufactured using a foundry bulk CMOS 90-nm low standby power (LP) process. This thesis presents the post-silicon validation results of these designs.

Chapter 1 gives an overview of the radiation effects and the test chip 24. It also talks about the pre-silicon and post-silicon validation methodologies used for the test chip. Chapter 2 presents the architecture of the eFlash and the test bench, and the validation results of the eFlash. Chapter 3 discusses the architecture of the HERMES design and test bench, and later explains the debug results. Chapter 4 gives a summary of the post-silicon validation results for the HERMES, eFlash, and test structures.

ACKNOWLEDGMENTS

I would like to thank my family for their unwavering support throughout my master's studies.

I would also like to thank Dr. Clark for giving me this post-silicon validation opportunity, Dr. Holbert for guiding me in the group meetings and Dr. Brunhaver for taking his time to serve as a committee member. I am also indebted to my colleagues Chad, Aditya, Chandru, Vinay, Yitao, Sai Chaitanya and Punit for their invaluable contributions, discussions, and support during the projects. I must also thank the graduate advisors Toni, Sno, and Lynn for their help with all the administrative procedures. Finally, I would like to thank SpaceMicro for funding this research.

TABLE OF CONTENTS

Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

CHAPTER 1. INTRODUCTION

1.1. Radiation Effects Overview

Radiation particles such as neutrons, protons, alpha particles or heavy ions strike the sensitive nodes in electronic circuits resulting in malfunctions. With the scaling of supply voltages and transistor sizes in modern technologies, electronic circuits used in the aerospace and commercial designs are becoming more vulnerable to radiation effects. Radiation hardening is a technique used for fabrication and design of electronic systems to withstand these radiation effects. Total ionizing dose (TID) [Barn06] and single event effects (SEEs) [Mavis02] are the two major radiation effects on MOS devices.

When a high-energy particle travels through a semiconductor, it leaves an ionized track behind, and this ionization may cause a localized effect. This effect is called a single event effect (SEE), and it may cause a glitch in a circuit output, or a bit flip in a memory or register. Destructive SEE effects (hard-errors) are permanent. The non-destructive soft-errors consist of single-event upsets (SEU) and single-event transient (SET). SEUs are an important type of SEE that affect the electronic systems. An SEU occurs when a single ion interacting with the chip causes a state change of a memory or register bit. This does not cause lasting damage to the device. However, if the device is not able to recover from the error, it may cause a malfunction or silent data corruption. Recently, SETs are becoming the primary cause of malfunction in several space applications [Koga93, Ecof94]. An SET happens when there is a voltage transient due to the charge collected from an ionization event in a non-state logic circuit, in the form of a spurious signal traveling through a circuit. These become soft-errors if captured by a subsequent latch.

Total ionizing dose (TID) deposits charge into the CMOS devices and thus affects them in multiple ways [Barn06]. These are threshold voltage (Vth) shifts, noise, mobility and leakage. The importance of TID effects is declining with the voltage scaling in logic devices because these effects are negligible at the low operating voltages and thin oxides. The TID effect (the threshold voltage shift) is negligible at thin oxides because it is proportional to the square of the gate oxide thickness. However, TID effects are significant in flash memory because floating gates have thick oxides and charge pumps operate at very high voltages. TID results on the embedded flash are discussed in CHAPTER 2.

1.2. Test Chip 24 Overview

1.2.1. TC24 Projects

Test chip 24 (TC24) was manufactured using a 90-nm low standby power (LSP) bulk CMOS process from TSMC. TC24 has three separate projects (Fig. 1.1) on one chip because including three projects in one die utilizes the area and effort more effectively. The three projects are the Highly Efficient Radiation-Hardened Microprocessor for Enabling Spacecraft (HERMES); embedded flash (eFlash); and some radiation-hardened by design (RHBD) flip-flop test structures. Input pads are shared across these three projects, and outputs from these three projects are multiplexed to output pads by using 2-bit block select signal (Blk_Sel<1:0>). The XOR clock multiplier provides the multiplied version of the phase shifted input clocks to the HERMES. The eFlash and the test structures get clocks from their respective input pins. The eFlash block has extra power pins for flash memory and charge pump.

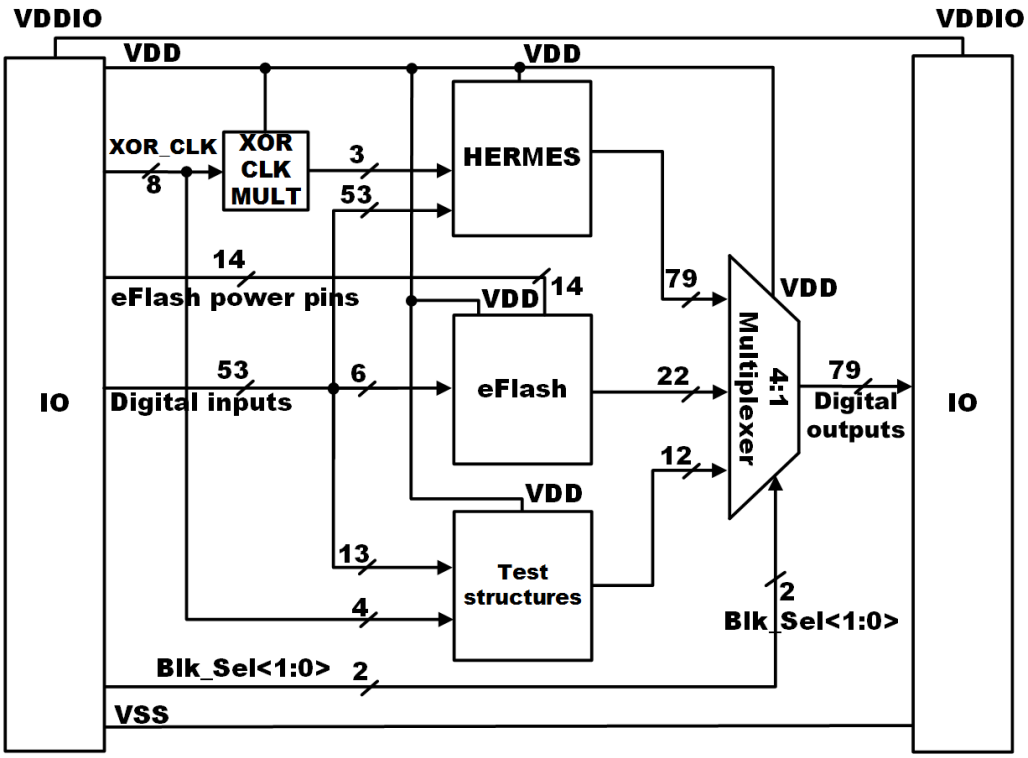


Fig. 1.1 TC24 top-level block diagram.

The HERMES (CHAPTER 3) block was designed to be a radiation-hardened microprocessor for space applications. It is a 32-bit MIPS32 4Kc core based processor [MIPS00], which incorporates multiple radiation hardening techniques at the transistor, layout, and micro-architecture levels. The MIPS32 4Kc architecture was used in the part due to the simplicity of the hardware implementation, and the MIPS does many operations in software. Although this processor core is intended for use as an embedded core in system-on-chip (SOC) designs, this implementation is in a stand-alone package for testing purposes.

The Xilinx XCFXXP series of FPGAs are programmed through a flash-based programmable read-only memory (PROM) chip. Hence, for the Xilinx FPGA to work in the radiation-hardened environment, the PROM should be radiation hardened. The eFlash

(CHAPTER 2) block is designed as a soft-error hardened flash-based PROM. To provide this function embedded flash memory block from Microchip Technology Inc. [Micro12] was used to implement the PROM portion of this chip. The eFlash chip is based on the Xilinx XCFXXP series of platform flash in-system programmable configuration PROMs.

1.2.2. Test Structures Overview and Beam Results

The test structures consist of five shift register chains, each 1024 bits long. These chains are numbered as chain 0, chain 1, chain 2, chain 3 and chain 4. Flip-flops in chain 0, chain 2 and chain 4 are temporally radiation hardened flip-flops and flip-flops in chain 1 and chain 3 are conventional unhardened flip-flops. These test structures were subsequently tested using a proton beam. To perform the testing, there are two test cases: static and dynamic. For static tests, alternating 0's and 1's are first shifted into the five flip-flop chains. Then clocks are gated to all the flip-flops, and the proton beam is turned on for 5 minutes. The data are then shifted out of the chains to see if any SEU errors have occurred. For dynamic tests, alternating 1's and 0's are continuously shifted into all five chains while clocking to all the flip-flops. This test is sensitive to both SEU and SET errors.

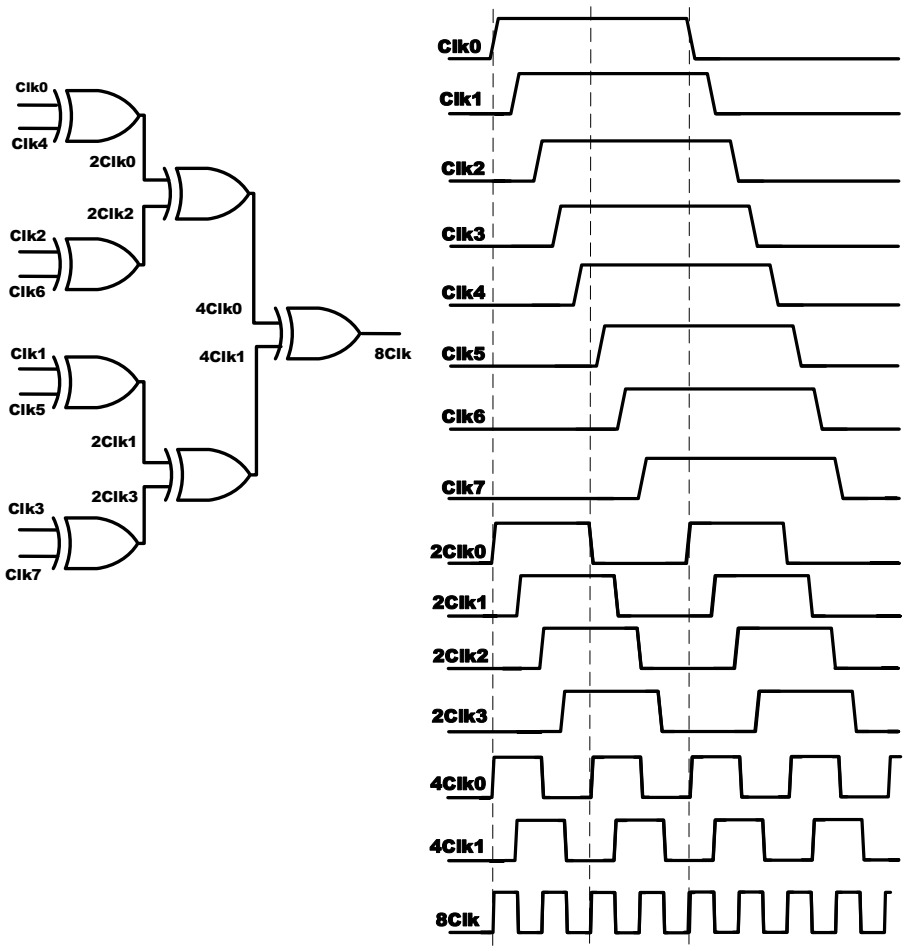
The proton beam testing was done on the test structures at the UC Davis cyclotron, and the results are summarized in Table 1-1. Versions d2, d3, d4, d7, d8, and d9 are dynamic tests, and they are operated at 40 MHz clock frequency. Versions s5 and s6 are static tests, and the data shifting is performed at 20 MHz clock frequency. Both the static and dynamic tests were operated at 0.9 V VDD, the lowest voltage possible without any timing errors. Errors were observed (highlighted in gray) only in flip-flop chains 1 and 3 because they are not radiation-hardened flip-flops. Since the test structures area was too small, a maximum of three errors were observed in the time that was allotted at the beam.

Voltage (V)	Flux	Fluence	Version	Frequency (MHz)	Static/ Dynamic	Total Errors	Errors in flop chain [4]		Errors in flop chain [3]		Errors in flop chain [2]		Errors in flop chain [1]		Errors in flop chain [0]	
							1->0	0->1	1->0	0->1	1->0	0->1	1->0	0->1	1->0	0->1
0.9	1.25E+07	1.00E+09	d2	40	Dynamic	0	0	0	0	0	0	0	0	0	0	0
0.9	1.27E+07	3.83E+09	d3	40	Dynamic	1	0	0	0	0	0	0	1	0	0	0
0.9	1.29E+07	3.88E+09	d4	40	Dynamic	1	0	0	0	0	0	0	0	1	0	0
0.9	1.27E+07	3.87E+09	s5	20	Static	1	0	0	0	0	0	0	1	0	0	0
0.9	1.27E+07	3.83E+09	s6	20	Static	3	0	0	0	2	0	0	0	1	0	0
0.9	1.29E+07	4.50E+09	d7	40	Dynamic	0	0	0	0	0	0	0	0	0	0	0
0.9	1.33E+07	3.99E+09	d8	40	Dynamic	0	0	0	0	0	0	0	0	0	0	0

Table 1-1 Test structures proton beam results.

1.2.3. XOR Clock Multiplier

Since a phase-locked loop (PLL) is very vulnerable to upsets in beam testing, an XOR clock multiplier was used to generate higher frequencies [Gogula15]. The XOR clock multiplier (Fig. 1.2 (a)) derives a maximum of multiplied-by-8 clock frequency by using eight clocks that operate at a lower frequency. To generate multiplied-by-8 frequency, these eight clocks must be 22.5 degrees (180 degrees / 8) phase shifted relative to each other (Fig. 1.2 (b)). If the relative phase shift between input clocks is not exactly 22.5 degrees (can be due to clock jitter), then the output clock will not have exactly 50% duty cycle. To generate multiplied-by-2, only two clocks must have 90 degrees (180 degrees / 2) phase shift between them and the remaining six clock pins must be held constant. Similarly, to generate other multiples of the clock frequency, the input clocks should be phase-shifted accordingly. For testing HERMES, these phase-shifted input clocks are generated from the test bench (mapped inside an FPGA).



(a) *(b)*

Fig. 1.2 (a) XOR clock multiplier logic diagram. (b) XOR clock multiplier waveform.

1.3. Pre-Silicon Validation

1.3.1. TC24 Test Bench Setup

To facilitate the use of same test bench at both the project and the chip level, project wrappers are created. Each project wrapper is created by assigning input pads of TC24 to respective project input names, output pads of TC24 to respective project output names and setting all other pads as respective constant values. For example, to create the top-

level wrapper for HERMES (Fig. 1.3), Blk_Sel value is set to select HERMES, and all other package pins are assigned to HERMES names.

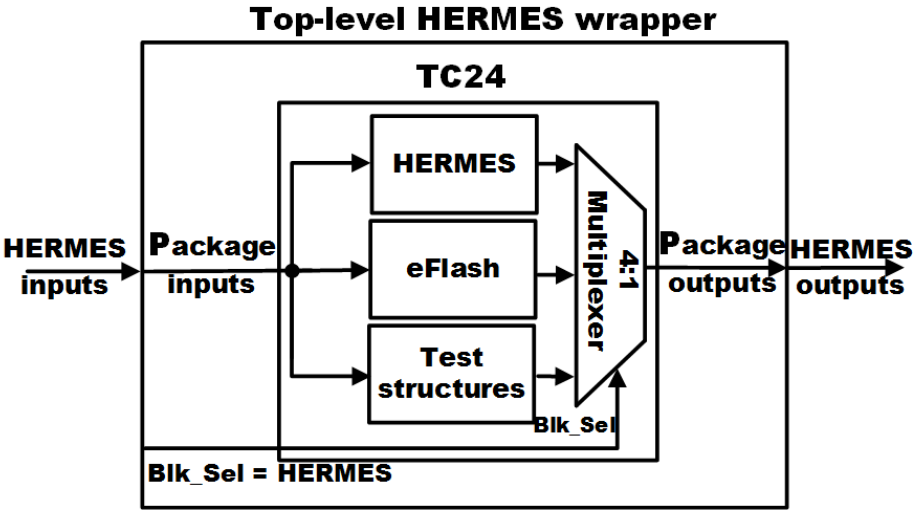


Fig. 1.3 Top-level HERMES wrapper for TC24.

In pre-silicon testing, the top RTL module instantiates a test bench (TB) and project wrapper as a device under test (DUT). The TB drives stimulus to the DUT, and top module displays the appropriate responses from the DUT. For some projects, DUT inputs are driven by the TB based on the outputs from the DUT (e.g. HERMES in Fig. 1.4). If incorrect responses are displayed from the DUT, then either the TB or the DUT should be modified accordingly until the correct responses are viewed from the DUT.

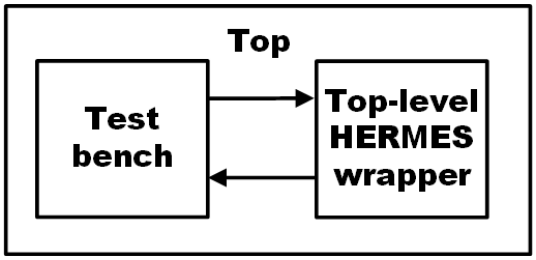


Fig. 1.4 HERMES pre-silicon testing setup.

1.3.2. Simulation Setup in ModelSim

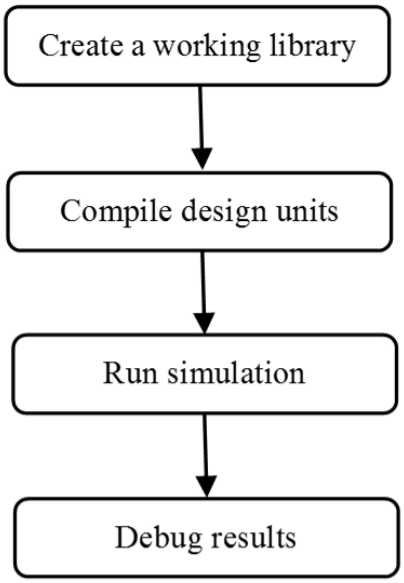


Fig. 1.5 Simulation flow in ModelSim [Mentor04].

There are four basic steps that need to be followed (Fig. 1.5) to simulate a test bench. The first step is creation of a working library. VHDL or Verilog files can be compiled into a single library in ModelSim. “Work” is the default destination library used by ModelSim for all the compiled designs. The second step is compilation of the design files. After creation of the working library, all the new compiled designs will be stored in the working library directory. The third step is to run the simulation. Invoke the simulator on a top-level test bench with the time resolution in nanoseconds, since the clock period is in nanoseconds. After successful loading of the design without any error, then the simulation can be started by entering a run command. The fourth step is to debug the results. The correctness of the design can be verified by viewing waveforms and timing sequence of the signals.

1.4. Post-Silicon Validation

1.4.1. TC24 Test Bench Setup

Post-silicon validation setup of the TC24 is shown in Fig. 1.6. The key components of the validation setup are a TC24 test bench, TC24 chip, and PC. The TC24 test bench is mapped to a Spartan 3 XC3S4000 FPGA. The ZestSC2board contains the FPGA, and it is attached to the TC24 chip using all the four IO pin arrays in the printed circuit board (PCB). The ZestSC2board is connected to the PC through USB 2.0. A C program inside the PC receives binary data from the USB and stores the data in a log file. The C program just records the data, and it does not control the FPGA.

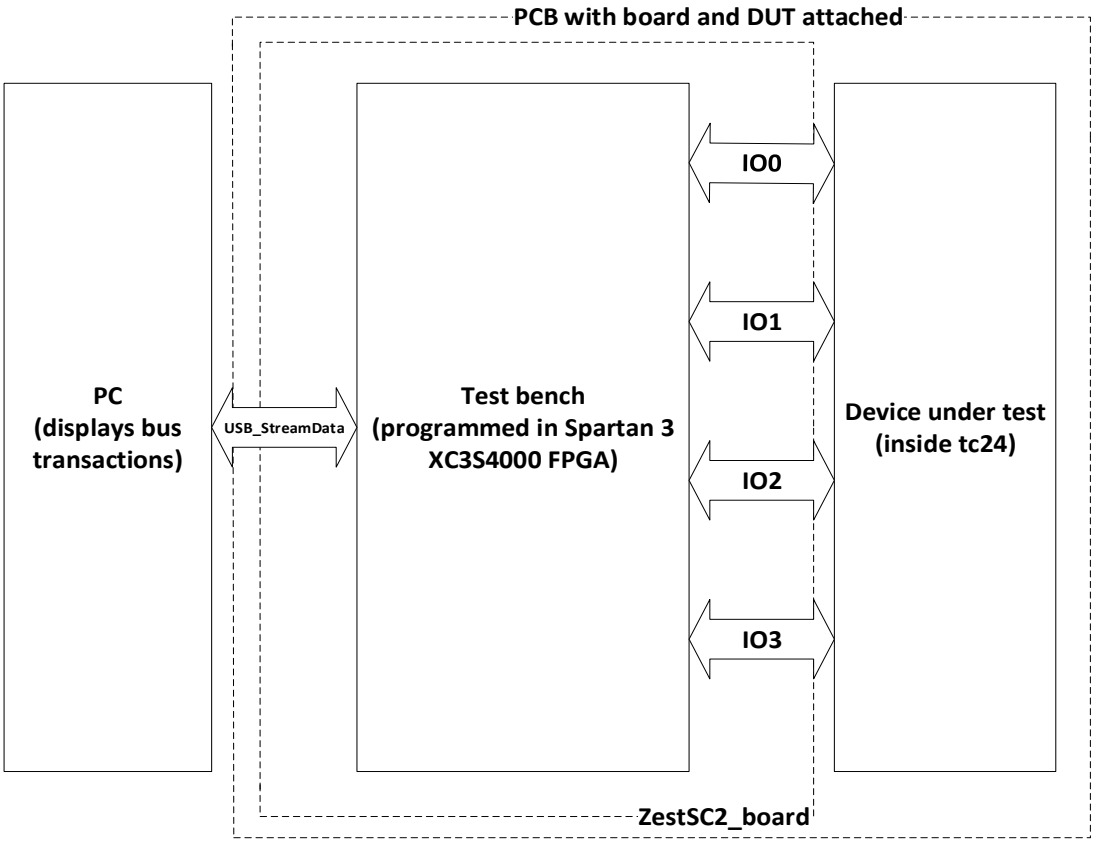


Fig. 1.6 Top-level architecture of the TC24 post-silicon validation.

The block diagram of the ZestSC2 board is shown in Fig. 1.7. It is a desktop board with a Xilinx Spartan-3 FPGA with up to four million system gates. The FPGA is connected to a PC using USB for configuration and data communication. It has four pitch headers that can be used for I/O connections. The primary method for data communication between the FPGA and USB is by the FX2's FIFO interface. This can operate at a frequency of 48 MHz with its data bus 16 bits wide giving a maximum burst data rate of 96 MBytes/sec. The USB streaming bus is bidirectional, and the bus reverses the direction for cycles specified by "User_StreamBusGrantLength".

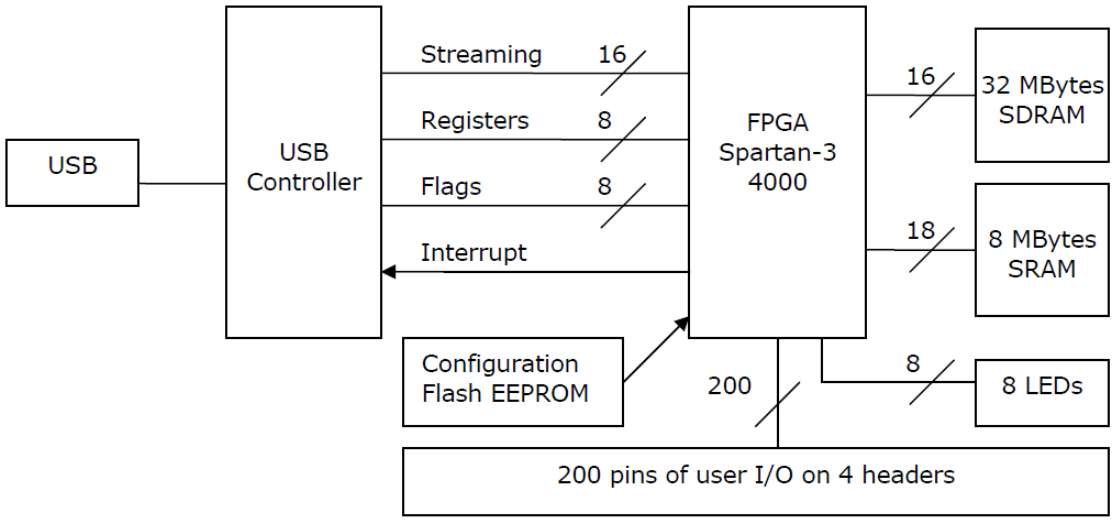


Fig. 1.7 ZestSC2 board block diagram [Orange10].

Orange Tree Technologies fabricated the ZestSC2board [Orange10] while all the IO-pins and the printed circuit board (PCB) were custom developed for the TC24. The ZestSC2 board is attached in a piggyback fashion to the PCB (Fig. 1.8) with the help of four IO jumper headers. The TC24 chip is attached to the custom PCB using a zero insertion force (ZIF) 289-pin socket (Fig. 1.9).

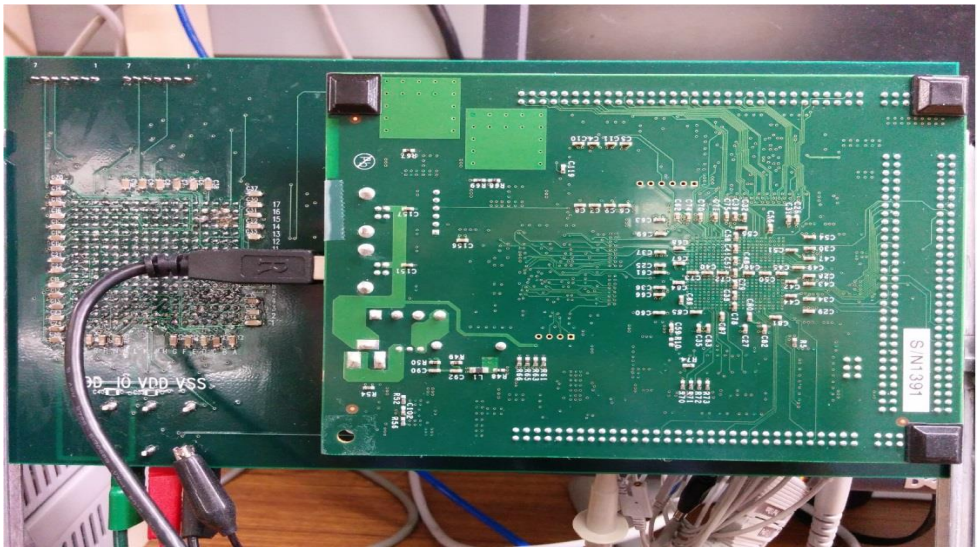


Fig. 1.8 ZestSC2 board attached to the PCB.

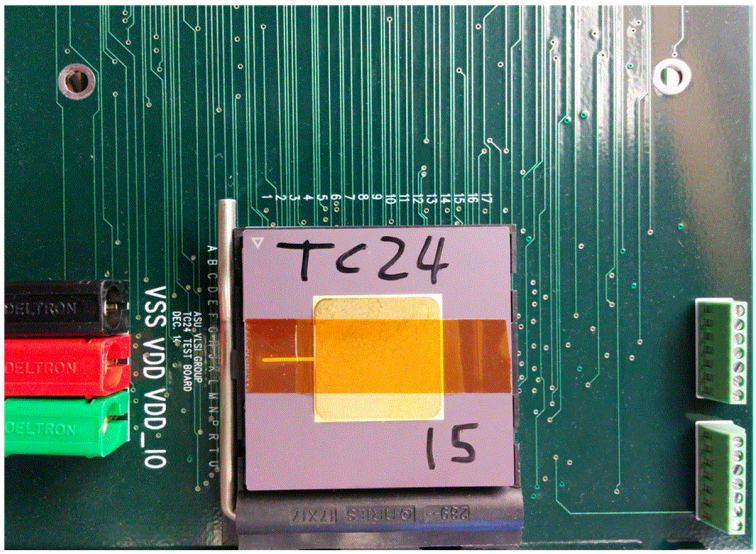


Fig. 1.9 TC24 (DUT) inserted in socket on custom PCB.

1.4.2. Post-Silicon Validation Setup in Other Universities

The paper [Carlo09] describes a low-cost FPGA based post-silicon validation architecture for static random access memories (SRAMs). They used a low-cost Xilinx ML-403 FPGA board, which is connected to the slave board (Fig. 1.10) through standard

expansion connectors. The slave board accommodates the SRAM device under test. Either Ethernet or USB can be used to communicate to the FPGA board. Instead of using BRAM inside the FPGA, they used Micro blaze processor on the FPGA. The Micro blaze processor is based on a 32-bit Harvard RISC architecture, and can access all external blocks and internal FPGA resources. They ran C programs on the Micro blaze processor to test the SRAM.

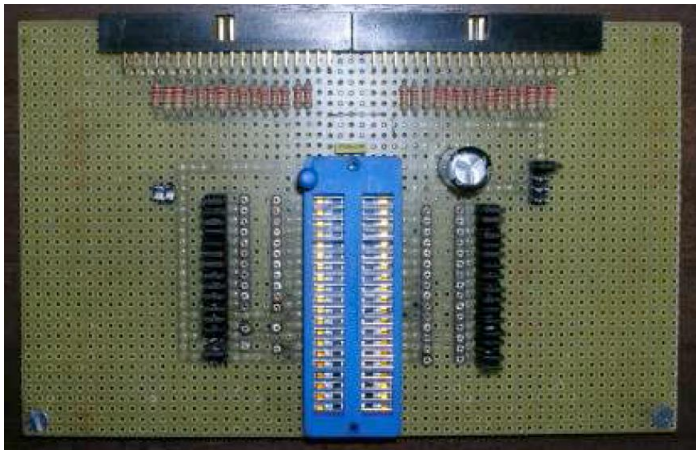


Fig. 1.10 Slave board for testing SRAMs [Carlo09].

The problem with the setup shown in Fig. 1.10 is that it can operate at only low frequencies (less than 40 MHz). However, this setup is not ideal to test the microprocessors as they operate at very high frequencies.

CHAPTER 2. EFLASH VALIDATION

2.1. eFlash Architectural Overview

The eFlash used in this study is designed to be a soft-error hardened controller and embedded flash for configuring Xilinx XCFXXP series of FPGAs (not the FPGA that was used for mapping test bench). A high-level block diagram of the RHBD eFlash is shown in Fig. 2.1.

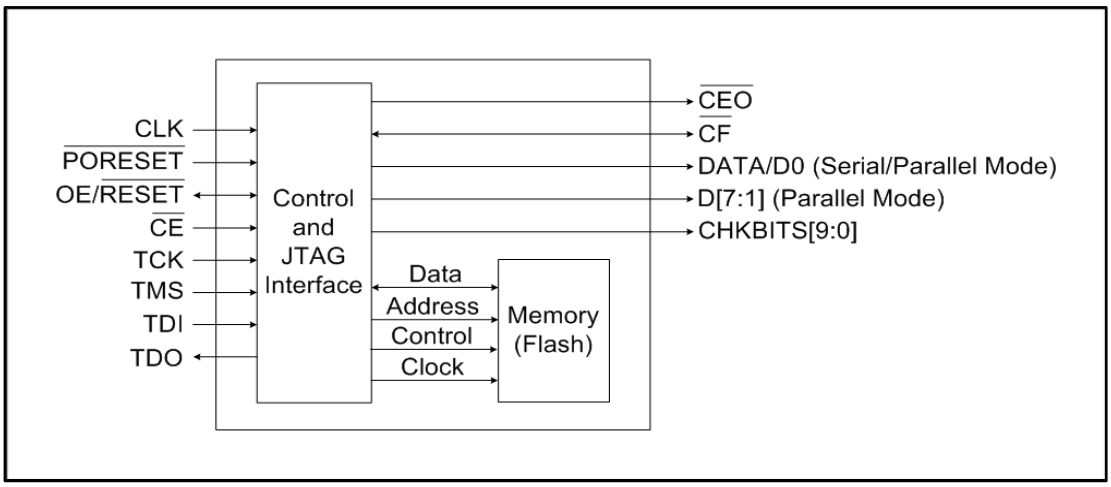


Fig. 2.1 RHBD eFlash block diagram.

As shown in Fig. 2.1, there are two major blocks in this device. The first block is the control and joint test action group (JTAG) interface. This block implements the JTAG test access port (TAP) controller [IEEE01], the flash memory controller and all other control functions required to interface with the flash memory and the external system. The second block is the flash memory. This block contains two instances of the embedded flash memory modules supplied by Microchip Technology Inc. [Micro12]. Only two embedded flash memory blocks are used to keep the eFlash size reasonably small and manageable from a design-time perspective. Each block can store 3.28 Mb of data and read/write

operations are performed using a 32-bit data interface. However, on this eFlash memory, only 16 of these 32 bits are used for data. Out of the remaining 16 bits, 10 bits are used as Bose Chaudhuri Hocquenghem (BCH) error correcting code (ECC) check bits [Naseer08] and 6 bits are unused. Since only half of the data available per block is used as configuration data, the total memory available for field programmable gate array (FPGA) configuration data is still 3.28 Mb.

This eFlash configuration leads to three vulnerabilities. Firstly, one of the two flash memory blocks could latch-up. This is a serious concern since high voltages are used when erasing or programming flash memory. Secondly, an SET could corrupt a critical control signal going to one of the two flash memory blocks during operation, resulting in an uncorrectable (and possibly undetectable) error. Thirdly, due to the layout of signals within the flash memory block, it is possible that a multi-node upset could occur, resulting in an uncorrectable (and possibly undetectable) data error occurring either on the way into the block (i.e., programming/erasing) or on the way out of the block (i.e., reading). Ideally, for RHBD based eFlash the signal locations are controlled. However, in this case, the eFlash memory layout is not RHBD.

2.2. Background for the Test Bench Setup

2.2.1. Flash Memory Map and Command Sequencing Related Registers

There are four sections (Table 2-1) in the flash memory block supplied by Microchip [Micro12]. The first section is Info Rows 0 (IFR0). IFR0 in each flash memory bank holds 1024 32-bit words. This portion of the flash memory contains circuit adjustment parameters and manufacturing information. The second section is Info Rows 1 (IFR1). IFR1 stores the user controlled registers (UCRs). The third section is main memory 0. Main

memory stores the FPGA configuration data and is split into two halves that do not have a contiguous address space. Main memory 0 is the lower half, which contains 53248 32-bit locations. The fourth section is main memory 1. Main memory 1 is the upper half of the main memory, which contains 54272 32-bit locations. More details on how to address each section of the eFlash memory are discussed in step 1 of section 2.2.3.

Number	Section of Memory	Address Range
1	IFR0	0x3FF-0x000
2	IFR1	0x103FF-0x10000
3	Main Memory 0	0xCFFF-0x0000
4	Main Memory 1	0x1D3FF-0x10000

Table 2-1 Flash memory map.

The command sequencing (described in 2.2.3) makes use of three TAP registers [Patt13] (Fig. 2.2). The first one is SEQ-INST register, which holds instructions for all four sequencers. The second one is SEQ-INFO register, which contains additional address/data information instructions. The final one is SEQ-PTR register, and it includes entry point vectors into the SEQ-INST registers for three of the sequencers. These registers (Fig. 2.2) are used for the command sequences associated with the three types of program and erase operations, as well as for the power-on-reset command sequence.

SEQ-INST register holds the commands that are to be executed. This is an 889-bit register that is selected and loaded using SSC_SEQ_INST instruction. This register is partitioned into 127 7-bit instruction entries. Each of these entries holds instructions executed within the instruction sequencer, which is invoked for each of the aforementioned four sequence types. For some of these instructions, four of the bits are used as a pointer that selects one of the 16 sequencer information registers in SEQ-INFO. The sequencer information registers provide additional information for the instruction.

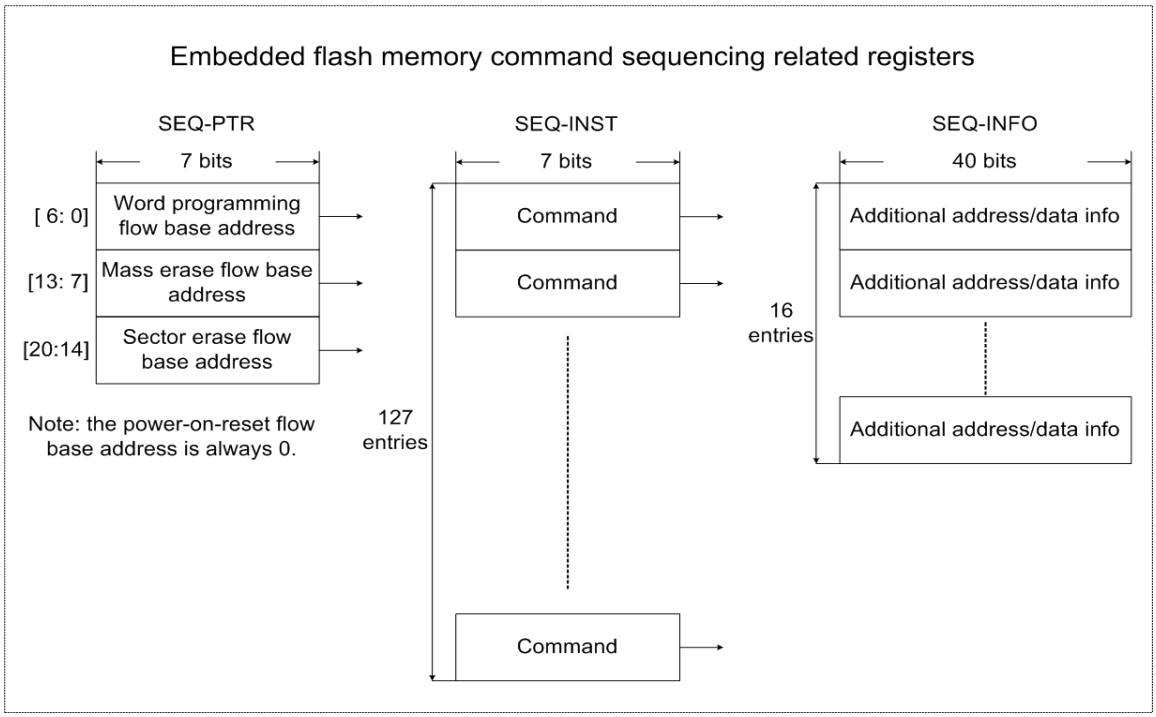


Fig. 2.2 Embedded flash memory command sequencing related registers.

SEQ-INFO is a 640-bit register that is selected and loaded using the SSC_SEQ_INFO instruction. This register is partitioned into 16 entries, with each entry corresponding to a 40-bit instruction sequencer information registers. Each of these registers holds address and data information for recall reads/loads, trim register writes, and test mode register writes.

Finally, SEQ-PTR is a 21-bit register that is selected and loaded using the SSC_SEQ_PTR instruction. This register is partitioned into three entries, with each entry corresponding to a 7-bit instruction sequencer pointer base registers. These registers hold pointers to the entry point associated with its corresponding type of flow. There is one flow for sector erase, one flow for mass erase, and one flow for word programming. Whenever the flow for a particular type of sequence is activated, the associated base pointer is loaded

as the starting instruction pointer for that flow. The flow then proceeds sequentially from there until it ends. For the power-on-reset flow, the entry point is always at location 0.

The SEQ_INST, SEQ_INFO and SEQ_PTR register values are programmed as shown in Fig. 2.3 for the eFlash TID experiments 1 and 2. Out of the four flows power-on-reset, word program and sector erase flows are used for the eFlash TID experiments. Mass erase erases the complete main memory. Hence, mass erase is never used since the main memory 0 and the main memory 1 are erased independently for experiments 1 and 2. More details about the experiments are described in section 2.4 and section 2.5.

```
// Programmable instruction sequencer registers (SEQ-INST, SEQ-INFO, SEQ-PTR)
// Note: 1) The SEQ_INST register values shown here are reverse in direction (888 downto 0) than compared to the MAS (0 to 888)
//       2) The SEQ_INFO register values shown here are reverse in direction (639 downto 0) than compared to the MAS (0 to 639)
//       3) The SEQ_PTR  register values shown here are reverse in direction ( 20 downto 0) than compared to the MAS (0 to  20)

// SEQ-INST
parameter SEQ_INST_REG =
{{114{7'b0000000}},// Unconditional End's
   7'b0000000,    // Unconditional End            //+ Sector erase flow
   7'b0000010,    // Flash Erase                  //+
   7'b0000000,    // Unconditional End            //+ Mass erase flow
   7'b0000010,    // Flash Erase                  //+
   7'b0000000,    // Unconditional End            //+ Word program flow
   7'b0000011,    // Flash Word Program           //+
   7'b0000000,    // Unconditional End            //+
   7'b1010101,    // Recall Load, SEQ-INFO Ptr = 5   |
   7'b1010100,    // Recall Load, SEQ-INFO Ptr = 4   |
   7'b1010011,    // Recall Load, SEQ-INFO Ptr = 3   | Reset flow
   7'b1010010,    // Recall Load, SEQ-INFO Ptr = 2   |
   7'b1010001,    // Recall Load, SEQ-INFO Ptr = 1   |
   7'b1010000};   // Recall Load, SEQ-INFO Ptr = 0 //+

// SEQ-INFO
parameter SEQ_INFO_REG =
 {40'h0000000000, // SEQ-INFO(15)
  40'h0000000000, // SEQ-INFO(14)
  40'h0000000000, // SEQ-INFO(13)
  40'h0000000000, // SEQ-INFO(12)
  40'h0000000000, // SEQ-INFO(11)
  40'h0000000000, // SEQ-INFO(10)
  40'h0000000000, // SEQ-INFO( 9)
  40'h0000000000, // SEQ-INFO( 8)
  40'h0000000000, // SEQ-INFO( 7)
  40'h0000000000, // SEQ-INFO( 6)
  40'h000000000A, // SEQ-INFO( 5)
  40'h0000000009, // SEQ-INFO( 4)
  40'h0000000008, // SEQ-INFO( 3)
  40'h0000000004, // SEQ-INFO( 2)
  40'h0000000003, // SEQ-INFO( 1)
  40'h0000000002};// SEQ-INFO( 0)

// SEQ-PTR
parameter SEQ_PTR_REG =
  {7'b0001011,    // Sector erase flow base pointer = 11
   7'b0001001,    //   Mass erase flow base pointer =  9
   7'b0000111};   // Word program flow base pointer =  7
```

Fig. 2.3 Command sequencing registers values used for the TID experiments 1 and 2.

2.2.2. Programming the Flash Memory

The program operation in flash memory pulls down the required bits to logic ‘0’. The user initiates flash memory programming manually via the TAP port by using the ISC_PROGRAM [IEEE02] instruction. When this instruction executes, it will result in the specified portion of the flash memory being programmed with the user supplied data. There are different methods of programming, and the direct access method was used for TID validation of eFlash to collect more data (error bits) by using all 32-bits in each bank.

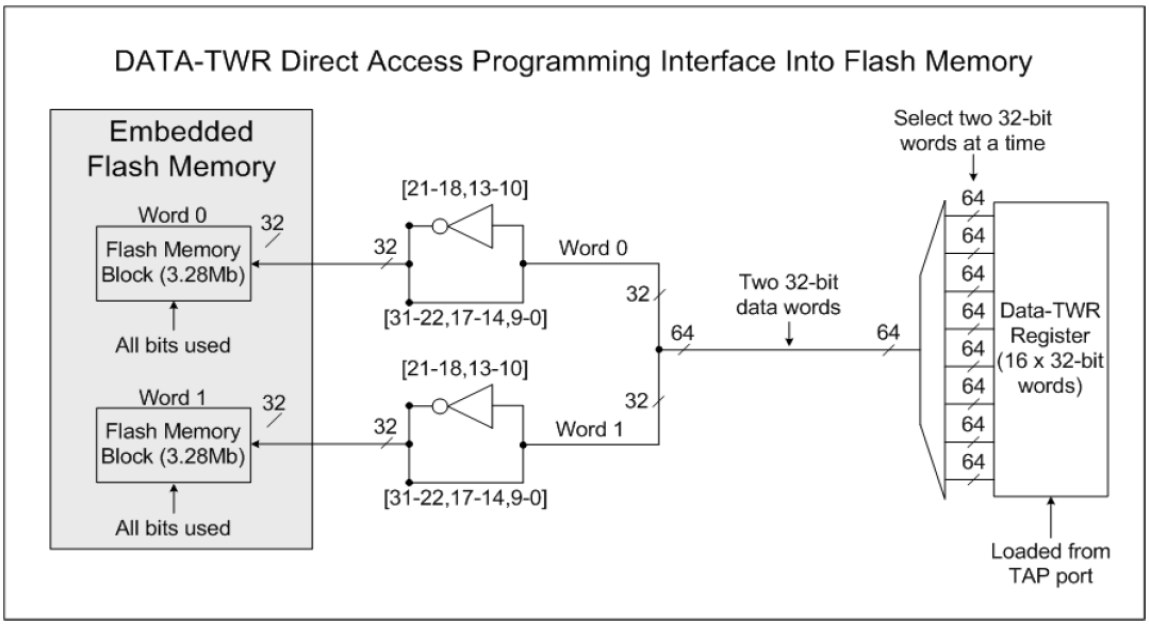


Fig. 2.4 DATA-TWR direct access programming interface into the flash memory.

In the direct access programming mode data are supplied into the 512-bit DATA-TWR register. This register bypasses the ECC check bit computations allowing direct programming of all bits in the flash memory. This allows better visibility into the eFlash errors. Eight writes will occur from DATA-TWR, each one programming 64 bits of data at a time, with the 32 LSBs going to flash bank 0 and the 32 MSBs going to flash bank 1. The first pair of 32-bit data words to be programmed will come from DATA-TWR[63:0],

the next pair from DATA-TWR[127:64] and so on. The path from DATA-TWR register to the flash memory is shown in Fig. 2.4.

Power Strip Connector (J1)	Power Strip Connector (J2)	Signal	Type	Description from MAS
	1	IN20U_0	Power	Embedded flash memory bank 0 20µA positive TC reference sink current, from pin to ground. It is used to generate a positive temperature analog output in HV regulator during the program operation.
7		IN20U_1	Power	Embedded flash memory bank 1 20µA positive TC reference sink current, from pin to ground. It is used to generate a positive temperature analog output in HV regulator during the program operation.

Table 2-2 20 µA current sources are required for the program operation to work.

To make the program operation work, current sources are needed to drive 20 µA currents on the pins IN20U_0 and IN20U_1 (Table 2-2). In a Microchip device, a bandgap circuit is used to control a precision current source on a die. To simplify the test setup, instead of using a current source, the same amount of current is generated by connecting the pins to ground through a resistor. Since, current sources cannot be shared, each pin should get independent current. The voltage measured at each pin is 3 V (Fig. 2.5) with the charge pump power supply at 3.3 V. Hence, after few calculations two 128 kΩ (3 V / 20 µA) resistors are connected separately to each pin.

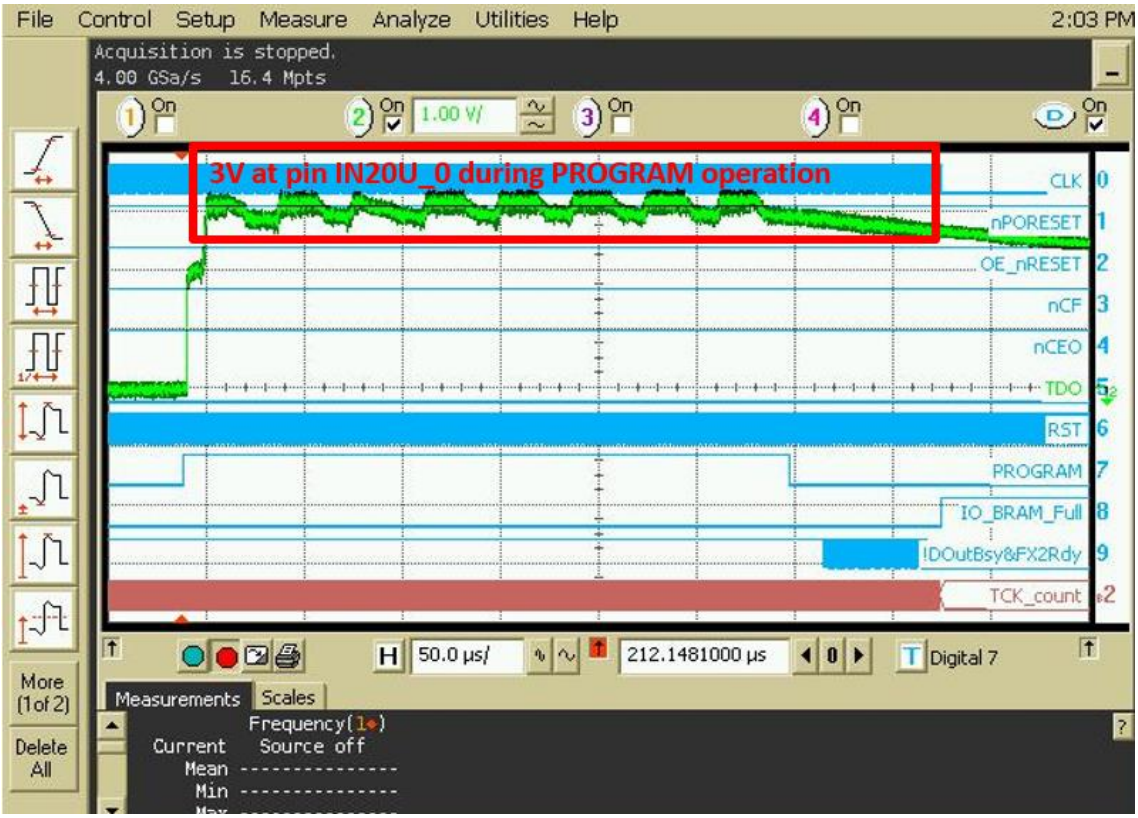


Fig. 2.5 Voltage measurement at pin IN20U_0 during the program operation.

To verify the program operation across multiple parts, part numbers 15 and 19 were chosen randomly. Many experiments were performed on parts 15 and 19 to determine the range of resistor values to be used. Parts 15 and 19 produced the same result for the programming operation at various charge pump voltages, so 128 kΩ resistor may work for all the parts. Table 2-3 lists various charge pump voltages at which 128 kΩ resistors work for the program operation. The charge pump voltage used for the eFlash TID experiments was 2.5 V (the row is highlighted in gray). The voltage was chosen as 2.5 V since it generates 20 µA, and allowed usage of the same power supply for the controlling FPGA.

VDD18 (V)	Resistor (kΩ)	Calculated Current (μA)	Programming Succeeded?
1.8	128	14	Yes
	∞ (open)	0	No
2.0	128	15.6	Yes
	∞ (open)	0	No
2.5	128	20	Yes
	∞ (open)	0	No
3.0	128	23.4	Yes
	∞ (open)	0	Yes
3.3	128	26	Yes
	∞ (open)	0	Yes

Table 2-3 Summary of the program operation with 128 kΩ resistor.

The main memory 0 starts at address 0x0000 and ends at address 0xCFFF while the main memory 1 starts at address 0x10000 and ends at address 0x1D3FF. The entire main memory can be programmed after initialization of the address register with the start address of the main memory 0. After initialization of the start address of the main memory 0, for each program operation, the address register increments from the start address of the main memory 0 to the end address of the main memory 0 in 3408000 TAP clock cycles. Subsequently, the address register rollovers from the end address of the main memory 0 to the start address of the main memory 1.

2.2.3. Erasing the Flash Memory

The erase operation pulls up all flash memory bits to logic '1'. The user initiates a flash memory erase manually via the TAP port by using either the ISC_ERASE [IEEE02] or the SSC_SECT_ERASE instruction. To simplify the setup for the eFlash TID experiments, only the SSC_SECT_ERASE operation was used to independently erase the

main memory 0 and the main memory 1. The procedure for using the SSC_SECT_ERASE instruction is as follows:

1. Load the ADDRESS register with the sector address to be erased. The upper two MSBs are used to specify the section of flash memory to be erased (00 = main memory, 01 = IFR1, 1X = IFR0). The next 10 bits are used to specify the sector address, and the four LSBs are ignored (these will be driven as 0s, as well as the three word address bits, to the flash memory during the sector erase operation).
2. Load the instruction register with the SSC_SECT_ERASE instruction.
3. Transition to the run-test-idle state for an amount of time sufficient to erase the sector.

Once a sector erase operation is initiated, it will be carried out to completion, regardless of whether the SSC_SECT_ERASE instruction that initiated the operation was executed in the Run-Test-Idle state for a sufficient amount of time (full sector erase takes 1011 μs to complete). This is done to prevent interrupting an erase operation, potentially resulting in the flash memory ending up in an indeterminate state. However, the TAP state machine must remain in the Run-Test-Idle state during the entire erase operation to avoid subsequent errors. One SSC_SECT_ERASE instruction erases 128 memory locations (128*32*2 bits = 1 KB).

The following steps should be followed to program/erase the eFlash [Patt13]:

1. Assert power-on-reset (nPORESET) initially to reset the chip. Following this, the chip will be idle
2. Program the SEQ_INST, SEQ_INFO and SEQ_PTR registers with the values shown in Fig. 2.3 via the TAP port

3. Set bit 11 of the DEBUG-CTL register. This has the effect of initiating the power-on-reset sequencer flow. Once the flow is complete, an FPGA configuration sequence will be automatically initiated. The latter can be subsequently re-initiated without going through the power-on-reset sequencer flow by simply causing some other form of reset to occur
4. Program and erase operations may now be performed and their corresponding sequencers can be re-programmed if desired. If a new power-on-reset command flow is desired, this sequence should be repeated starting at step (1) above

2.2.4. Reading the Flash Memory

The entire flash memory, or any portion thereof, can be read out via the TAP port by executing the XSC_READ instruction. This results in one bit of data being shifted out on TDO per TAP clock (TCK). Normal read, HAGE read with VREAD0 asserted, and VREAD1 asserted are three types of read operations on the eFlash memory. To simplify our setup, only the normal read was used for the eFlash TID experiments, and the procedure for doing the normal read operation is:

1. Load the DEBUG-CTL register with the desired read mode (bits 13-12 are used to specify the desired mode): DEBUG-CTL [13:12] = 00 => normal read. The state of these bits coming out of a power-on-reset or TAP reset is "00", so this step can be bypassed for a normal read in this situation.
2. Load the instruction register with the ISC_ADDRESS_SHIFT instruction.
3. Load the ADDRESS register with the starting 256-bit page address to be read out.
4. Load the instruction register with the XSC_READ instruction.
5. Transition to the Run-Test-Idle state and remain there for at least four TAP clocks.

6. Transition to the Shift-DR state and remain there for as many TAP clocks as required to shift out the bits desired. This can be for any number of bits, up to the total number of bits comprising either the main, IFR1 or IFR0 sections of flash memory. Since the three sections of flash memory are not contiguous, the ADDRESS register must be reloaded when transitioning from one section of memory to the other.

2.2.5. TBTest25 Test Case Run in ModelSim

The block diagram of the eFlash test bench (synthesizable) setup is shown in Fig. 2.6. With this scheme, depending on the test case, states can be added or removed from the state machine to provide necessary steps for the experiment. Moreover, each state is internally a state machine. Each internal state machine has a clock counter. The START input signal initiates the clock counter, and once the clock counter's count is finished, the internal state machine asserts the DONE output signal. The internal state machine gives the TMS and TDI outputs based on its clock count value.

First, the test logic reset sequencer state is initiated by the reset (RST) signal. After the reset sequencer's clock counter count is finished, the DONE signal is asserted which triggers the START signal of the next state in the state machine. Similarly, the complete state machine is traversed.

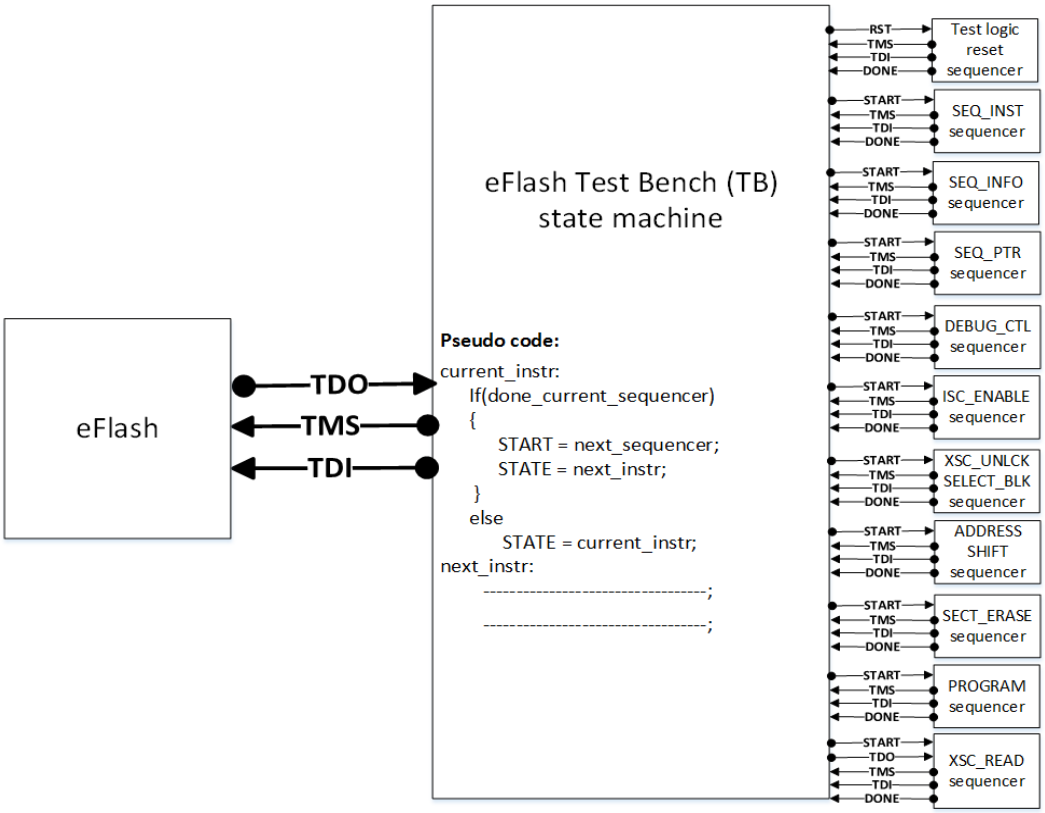


Fig. 2.6 Block diagram of the eFlash synthesizable test bench.

The TBTest25 test case contains five steps. First, read the complete main memory 0 using the XSC_READ operation to observe the stored data. Second, program all 0's into the full main memory 0 using the ISC_PROGRAM operation. Third, read the complete main memory 0 using the XSC_READ operation to verify the correctness of the programmed data. Fourth, erase the complete main memory 0 using the ISC_ERASE instruction to initialize all the flash memory data as 1's. Finally, read the full main memory 0 using the XSC_READ operation to verify the previous erase operation. In ModelSim (pre-silicon validation), the memory values are initialized as all 1's because the flash memory behavioral model initializes all 1's. Hence, the program operation was performed before the erase operation. However, for post-silicon validation, the program (second step)

and the erase (fourth step) operations were swapped. Since the program operation can only be done after the erase operation.

Fig. 2.7 ModelSim run on the TBTest25 test case.

Fig. 2.7 shows the ModelSim waveform output of the TBTest25 test case. The program operation writes eight double words in the eFlash memory. This operation takes 250 µs to complete and the recommended time is 224 µs. In the read operation after the program operation, at address 0x00006, the read data are equal to the programmed data (0xEED2D2EE). Hence, the program and the read operations are verified. Each sector erase clears (logic '1's) 512 bytes of data in the eFlash memory. This operation takes 1228 µs to complete and the recommended time is 1011 µs. After the sector erase on the main memory 0, the read data is all 1's (0xFFFFFFFF). Therefore, the sector erase operation has worked properly.

2.3. Test Bench Setup on Gammacell Board

2.3.1. Post-silicon Validation Setup Overview

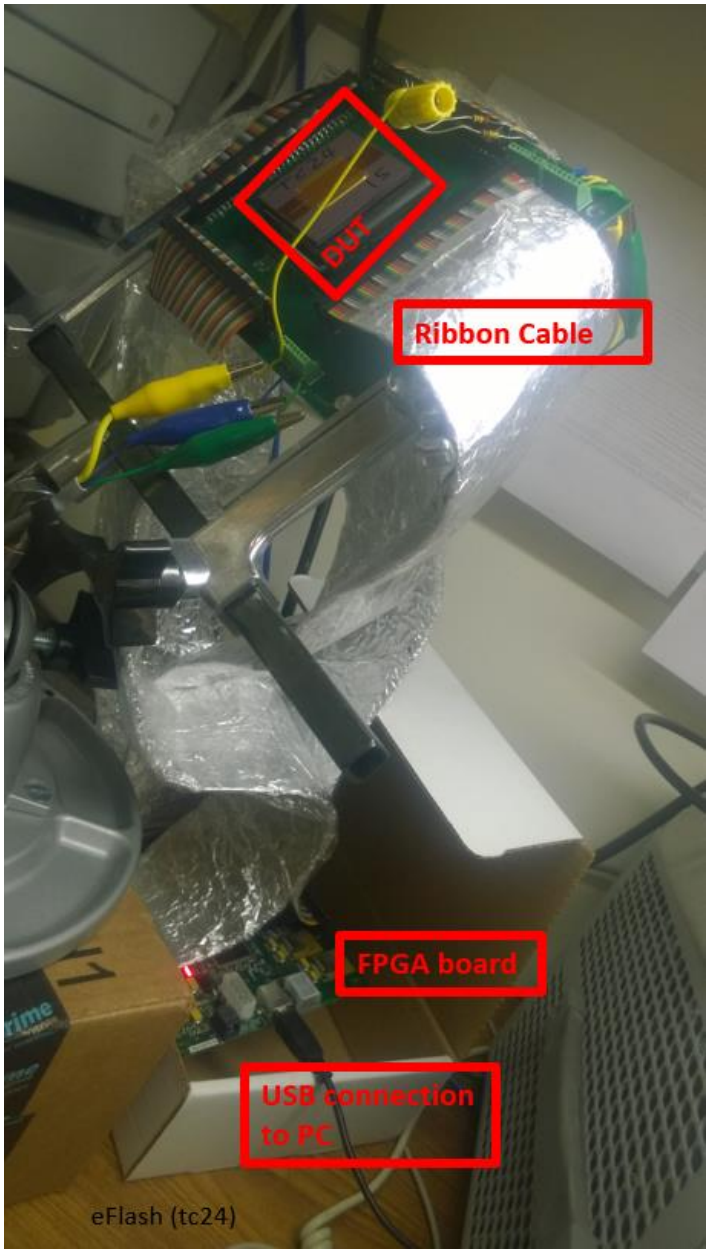


Fig. 2.8 Image of the eFlash post-silicon validation setup.

Fig. 2.8 shows a picture of the eFlash post-silicon validation setup on the Gammacell board. The TC24 chip (DUT) is attached to the Gammacell PCB board using a zero insertion force (ZIF) 289-pin socket. The Gammacell PCB board was designed for

the TID experiments, to go inside the Gammacell irradiator. The ribbon cables connect all four IO pin arrays from the Gammacell board to the ZestSC2board [Orange10]. The long ribbon cables are connected to place the ZestSC2board (FPGA board) outside the Gammacell irradiator. The ZestSC2board is connected to the PC via USB. The PC receives the data from the ZestSC2board and reports results in a log file.

2.3.2. Pull-up Resistors and Power Supply Connections

Signal	Type	Description
OE_nRESET	Open drain I/O	Output Enable/Reset (open-drain I/O with active low reset)
nCF	Open drain I/O	Configuration pulse (open-drain I/O, active low)

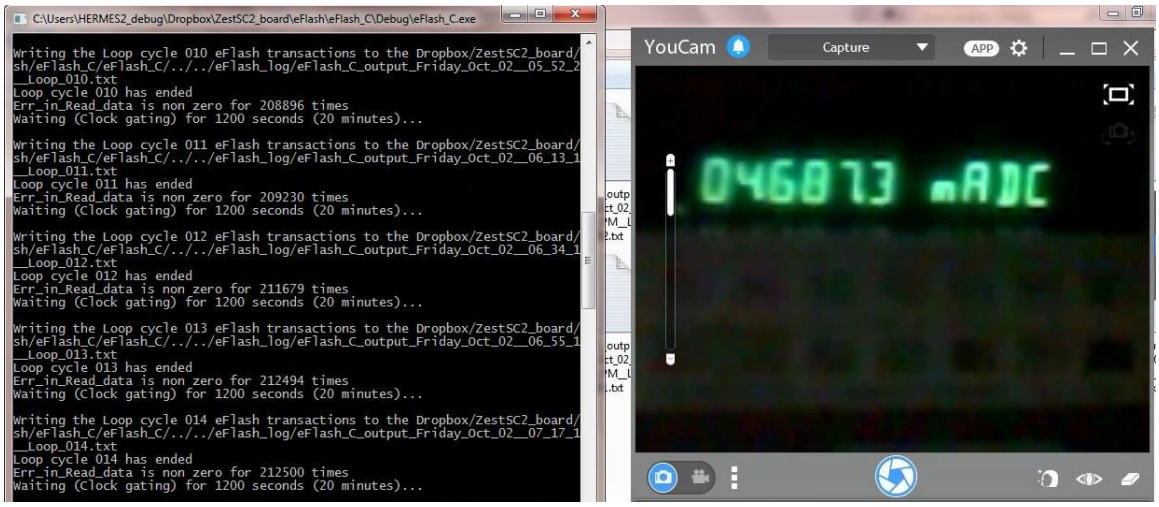
Table 2-4 OE_nRESET and nCF pins are open drain I/O's [Patt13].

The OE_nRESET and the nCF pins are open drain I/O's, as shown in Table 2-4. Since an open drain I/O pin needs a pull-up resistor, two 32 kΩ pull-up resistors are connected to the pins OE_nRESET and nCF. 32 kΩ resistors are used because the maximum current allowed to the eFlash pins is 100 µA, and the FPGA pins operate at 2.5 V (2.5 V /32 kΩ < 100 µA). The IO voltage pin VDDIO is connected to 2.5 V because the FPGA IO's operate at this voltage. The charge pump voltages (VDD18_0 and VDD18_1) are connected to 2.5 V because the 128 kΩ resistors generate exactly 20 µA at this voltage. The core voltages VDD_0, VDD_1 are connected to the maximum allowed voltage 1.32 V to avoid timing errors while reading the eFlash memory.

2.3.3. Measuring Current and Word Errors during the TID Experiments

The word errors (refer to section 2.4.3) from the C program inside the PC (Fig. 2.9 (a)) and the charge pump current (Fig. 2.9 (b)) were tracked throughout the experiment by logging in remotely to the PC in the room containing TID chamber. The charge pump

current was measured to observe the charge pump failure (leakage current) with respect to TID dose. The test bench waits for 20 minutes between each loop cycle (see Fig. 2.10), and prints the word errors (sum of word errors in both the main memory 0 and the main memory 1) that occur in each loop cycle.



(a) *(b)*

Fig. 2.9 (a) Word errors from the C program and (b) The charge pump current.

2.4. eFlash TID Experiment 1 Results

2.4.1. State Machine (Erase, Program, and Read Continuously)

The state machine "TBTest29_state_machine_150920_v07" was used for the eFlash TID experiment 1. Fig. 2.10 shows the pseudo code for the state machine:

```
Erase Main Memory 1;
Read Main Memory 1;
Program Main Memory 1;
Read Main Memory 1;
Read Main Memory 0;
For each loop cycle
{
        Erase Main Memory 0;
        Read Main Memory 0;
        Program Main Memory 0;
        (Invert the programming data
        for each new Program operation)
        Read Main Memory 0;
        Read Main Memory 1;
        Wait for 20 min;
}
```

Fig. 2.10 Pseudo code of the state machine used for the eFlash TID experiment 1.

For experiments 1 and 2, the main memory 0 and the main memory 1 are treated as two separate memory blocks. The main memory 1 is used to observe the read-only failures, and the main memory 0 is used to observe the program and erase failures. The erase operation pulls up all the flash memory bits to logic '1'. The program operation only pulls down the required bits to logic '0'. Hence, the erase operation has to be performed before the program operation. Coming to the main memory 1 in the state machine (Fig. 2.10), first, erase operation is performed on the complete main memory 1 and then program operation is performed on the main memory 1. In the program operation, all the locations of the main memory 1 are programmed with the same data as the address in the first loop. After erase and program operations, for each loop cycle, read operation is performed in the main memory 1 continuously. Coming to the main memory 0, for each loop cycle, erase, read, program, and read operations are performed continuously in the main memory 0. In the program operation, for odd loop cycles, the same data as the address are programmed

in the main memory 0. For even loop cycles, the inverted data of the address are programmed in the main memory 0. The test bench waits for 20 minutes between the each loop cycle to reduce the activity factor of the eFlash memory.

Address range 0x0000 to 0xCFFF in the main memory 0 and address range 0x10000 to 0x1CFFF in the main memory 1 are used for erase, program and read operations (highlighted with red colored borders in Table 2-5). To maintain consistency across read operations between the main memory 0 and the main memory 1, 0x1D000 to 0x1D3FF address range in the main memory 1 is not used.

Number	Section of Memory	Address Range
1	IFR0	0x3FF-0x000
2	IFR1	0x103FF-0x10000
3	Main Memory 0	0xCFFF-0x0000
4	Main Memory 1	0x1D3FF-0x10000

Table 2-5 Main memory 0 and main memory 1 were used for TID experiments 1 and 2.

The complete state machine used for experiment 1 is shown in Fig. 2.11. All TAP logic is reset when the TAP state machine enters the TEST_LOGIC_RESET state. SSC_SEQ_INST, SSC_SEQ_INFO and SSC_SEQ_PTR states program the sequencer instruction registers SEQ-INST, SEQ-INFO, SEQ-PTR respectively (refer to section 2.2.1). SSC_DEBUG_CTL_INSTR state sets bit 11 of the DEBUG-CTL register, which executes the (empty) power-on-reset flow. ISC_ENABLE_INSTR state enters the TAP state machine into ISC mode (this is required prior to erasing flash memory). XSC_UNLOCK_INSTR state unlocks Block 0 and GUCR (using in conjunction with SELECT-BLOCK register). This is required in order to erase a locked block of memory. TAP_IDCODE_INSTR state checks that the TAP state machine is in correct state and to make sure that the chip is working fine (output data will be 0xa5a5a5a5).

ISC_ADDRESS_SHIFT_INSTR state shifts the required start address into ADDRESS register for ISC_PROGRAM (program), SSC_SECT_ERASE_INSTR (sector erase) and XSC_READ (read) operations. SSC_DATA_TWR_INSTR loads the data to be programmed into DATA_TWR register.

Each state shown in Fig. 2.11 is internally a state machine. For example, TEST_LOGIC_RESET state has four internal states. These four internal states generate TMS values for the TAP controller inside eFlash to transition from TEST-LOGIC-RESET, RUN-TEST-IDLE, SELECT-DR-SCAN and SELECT-IR-SCAN states.

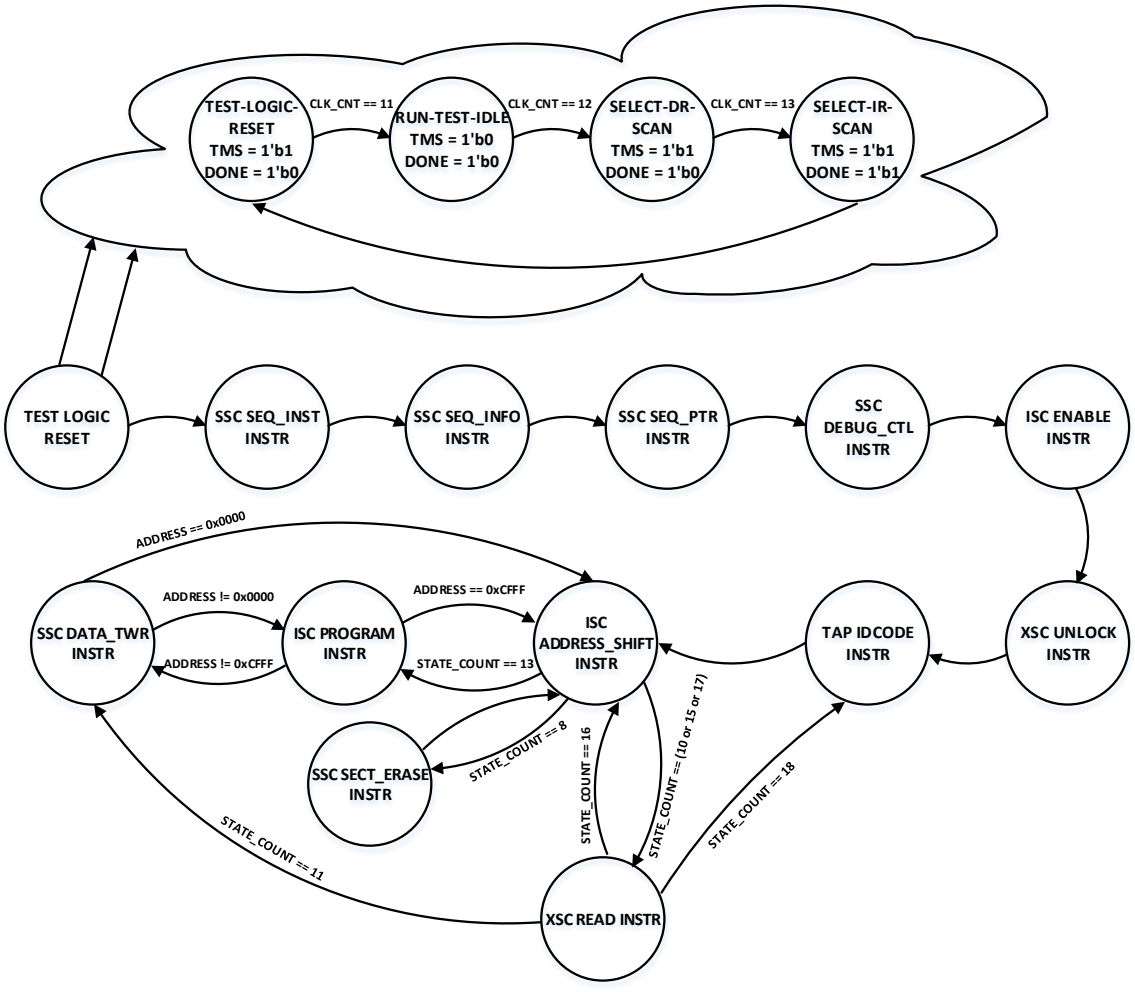


Fig. 2.11 State machine used for the eFlash TID experiment 1.

2.4.2. Output after 4 Hours of TID Exposure (90 krad)

The variables shown in Fig. 2.12 are sent through the streaming interface (USB) from FPGA to PC. The "clk_cnt" variable is a 16-bit clock count value of the TAP clock. It serves as a reference to check if all the transactions are recorded properly. The "control" variable indicates the status of reset and other control pins for the eFlash. The "current_instr" variable indicates the name of the current instruction executing in the TAP instruction register. After the read operation, the "Read_data_TDO" variable shows the 32-bit data given by the flash memory, and the "Err_in_Read_data" variable shows the XOR of the expected data and the actual data. The "loop-cnt" variable is an 8-bit counter of the current loop cycle (loop cycle is defined in Fig. 2.10). The output log file names are created based on this loop count value and the time stamp. The time stamps on the log files are used for total dose calculations. During the experiment, it takes exactly 30 seconds to write a log file for each iteration (loop cycle).

```
// Structure to recieve eFlash bus signals in one read operation
typedef struct{
        unsigned  int           clk_cnt : 16,    // 16-bit TAP clock cycle count [15:0]
                                loop_cnt: 8,     // 8-bit Loop count [7:0]
                                control : 8;     // Reset and other control pins for eFlash
        unsigned short int  current_instr;       // current-instr is 16-bits
        unsigned short int  Read_data_TDO_lsb;   // Read data lsb (16-bit)
        unsigned short int  Read_data_TDO_msb;   // Read data msb (16-bit)
        unsigned short int  Err_in_Read_data_lsb;// Err bits in Read data lsb (16-bit)
        unsigned short int  Err_in_Read_data_msb;// Err bits in Read data msb (16-bit)
}outdat;
```

Fig. 2.12 Streaming bits and their variable declarations in the C program.

The dose rate for the TID experiment 1 was 437 rad(Si)/min. The main memory 0, on which the read-only operations were performed, worked until 100 krad dose. However, the main memory 1, on which the erase, program and read operations were performed continuously, failed at 30 krad dose. The log file (Fig. 2.13), obtained after 4 hours of

exposure (4 * 60 * 437 = 90 krad), is discussed below to show that the erase operation failed earlier than the other operations. The first read operation (Fig. 2.13) after the erase operation on the main memory 0 gives mostly all 32 bit 0's instead of "ffc3c3ff" (erase failure). The expected read data after the erase operation is "ffc3c3ff" instead of "ffffffff" because of the inversion of bits in the read path (Fig. 2.4). The second read after the program operation on the main memory 0 gives mostly all 32 bit 0's instead of "ffffffff", because the previous erase operation failed to pull the bits to logic '1'. In the lower right corner of Fig. 2.13, the read-only operation on the main memory 1 gives the expected data. Hence, the conclusion was erase operation on the eFlash memory failed far ahead than other operations.

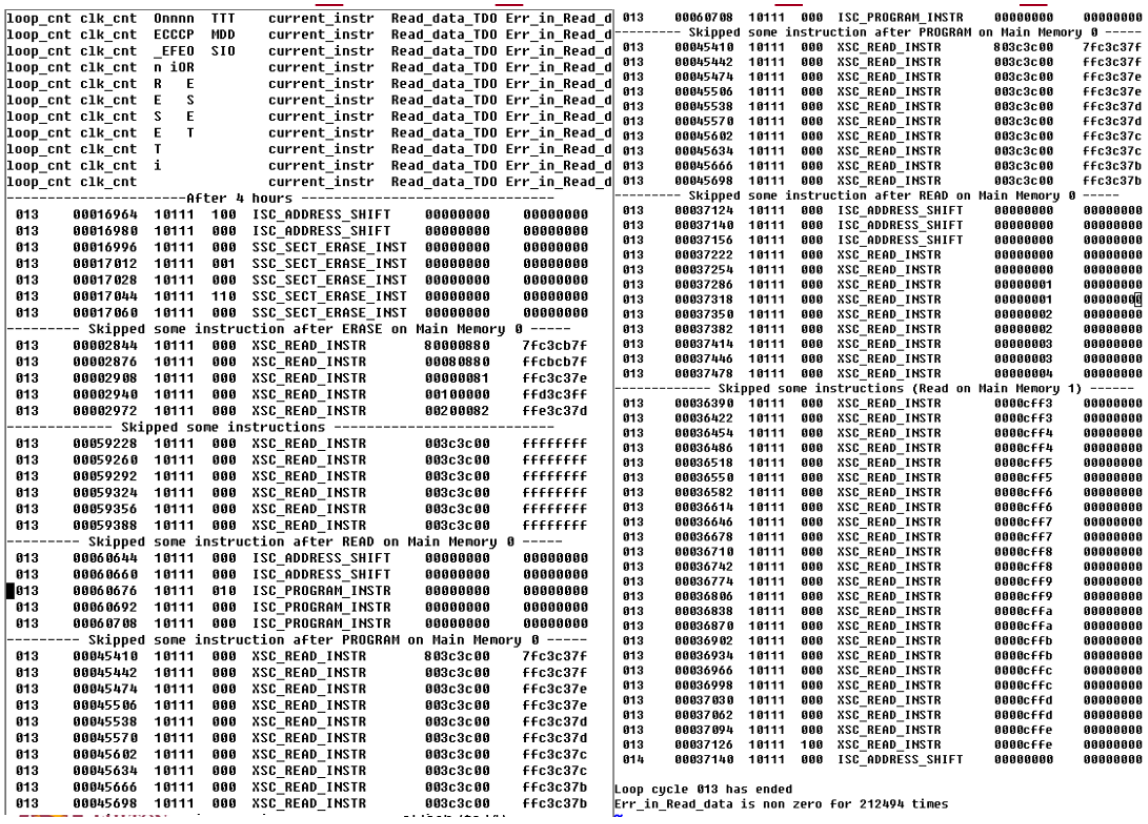


Fig. 2.13 C program output for the loop cycle 13.

2.4.3. Observations

As mentioned above, the main memory 0 has two banks with 53247 total locations. Therefore, the maximum possible number of word-errors for one read operation on the complete main memory 0 is equal to 53247*2. Since a read operation is performed after both erase and program operations in the main memory 0, the maximum possible word-errors is equal to 2*53247*2. Therefore, a maximum of 212988 word-errors (not bit-errors) can be observed in the main memory 0.

In the main memory 0, 212477 word-errors were observed, and the maximum possible word-errors are 212988. Therefore, 99.7% of possible word-errors were observed in the main memory 0. In the main memory 1 (continuous read operation on this main memory), 105796 word-errors were observed, and the maximum possible word-errors are 106494. Therefore, 99.3% of possible word errors were observed in the main memory 1. Thus, based on Fig. 2.14, it may be concluded that the erase operation failed below 50 krad, and the flash memory was completely non-functional after 100 krad.

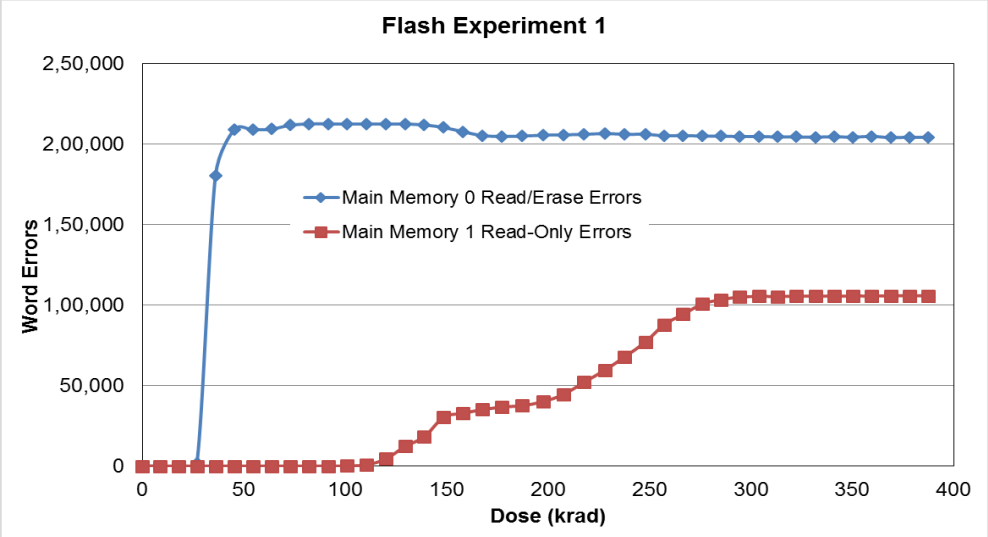


Fig. 2.14 Word errors observed in the eFlash TID experiment 1.

The same charge pump is used for both the erase and the program operations. Therefore, when the charge pump fails both the erase and the program operations will fail. Fig. 2.15 shows the bit-errors observed in experiment 1. One million bits failed at 36 krad dose for both the erase and the program 1 to 0 operations. Close to one million bits failed at 200 krad dose for the program 0 to 1 operation. For the program operation, bits predominantly fail in the 1 to 0 direction because of the failure in erase operation. The read-only operation started failing after 100 krad dose. Hence, the conclusion is that the erase operation failed before than the program and read operations.

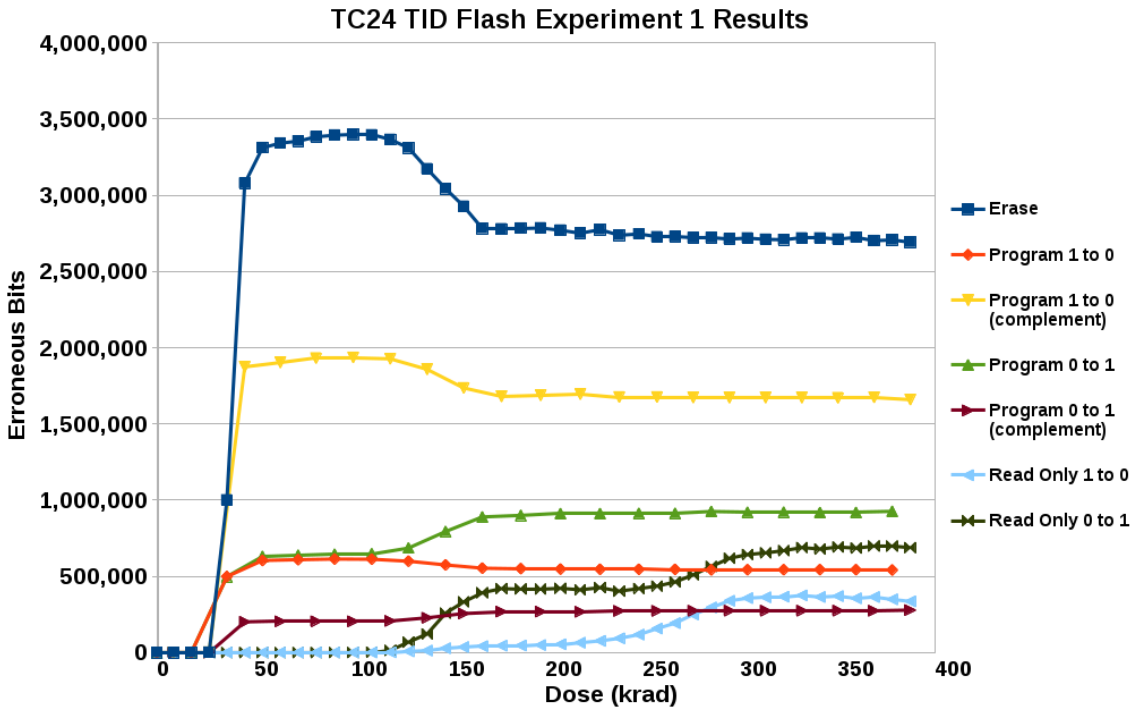


Fig. 2.15 Bit errors observed in the eFlash TID experiment 1.

As shown in Fig. 2.16, the current drawn by the charge pump constantly increased with an increase in the TID radiation dose.

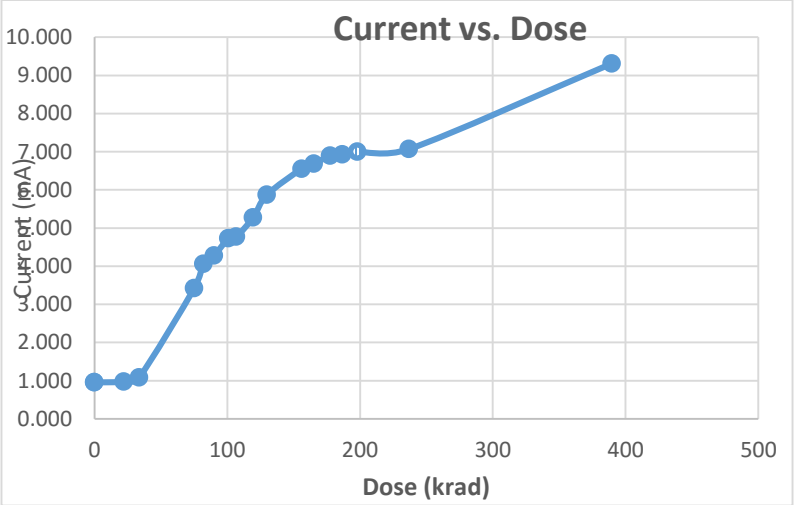


Fig. 2.16 Charge pump current vs. dose.

2.5. eFlash TID Experiment 2 Results

2.5.1. State Machine (Erase, Program, and Read Continuously)

In experiment 1, all the locations in main memory 0 are programmed with 32-bit data same as the value of each address location. The address range of the main memory 0 in hexadecimal is 0x0000 to 0xCFFF. Since the 16 most significant bits (MSBs) of the programmed data are always logic '0', a comparison could not be made between the number of 1 to 0 and 0 to 1 fails in the read operation. To observe the similar number of 1 to 0 and 0 to 1 fails, an equal number of 1's and 0's were programmed in the main memory 0 for experiment 2. The state machine "TBTest29_state_machine_151026_v09" was used for experiment 2. Pseudo code for the state machine is shown in Fig. 2.17:

```
Erase Main Memory 1;
Read Main Memory 1;
Program Main Memory 1;
Read Main Memory 1;
Read Main Memory 0;
For each loop cycle
{
        Erase Main Memory 0;
        Read Main Memory 0;
        Program Main Memory 0;
        (Refer to pseudo code in Fig. 2.18 for the programming data)
        Read Main Memory 0;
        Read Main Memory 1;
        Erase 1st sector in Main Memory 0;
        Wait for 20 min;
}
```

Fig. 2.17 State machine used for the eFlash TID experiment 2.

Only the programming data differ from the state machine used in experiment 1 (Fig. 2.10) to the state machine used in experiment 2 (Fig. 2.17). Programming data for the even loop cycles (Fig. 2.18) in experiment 2, are same as the address for the even locations and are the inverse of the address for the odd locations. Programming data for the odd loop cycles are same as the address for the odd locations and are the inverse of the address for the even locations.

```
if (Cycle %2 == 0)
{
        if (Address % 2 == 0)
                Program data same as Address
        else
                Program data inverse as Address (~Address)
} else
{
        if (Address % 2 == 0)
                Program data inverse as Address (~Address)
        else
                Program data same as Address
}
```

Fig. 2.18 Programming data in the main memory 0 and the main memory 1.

2.5.2. Observations

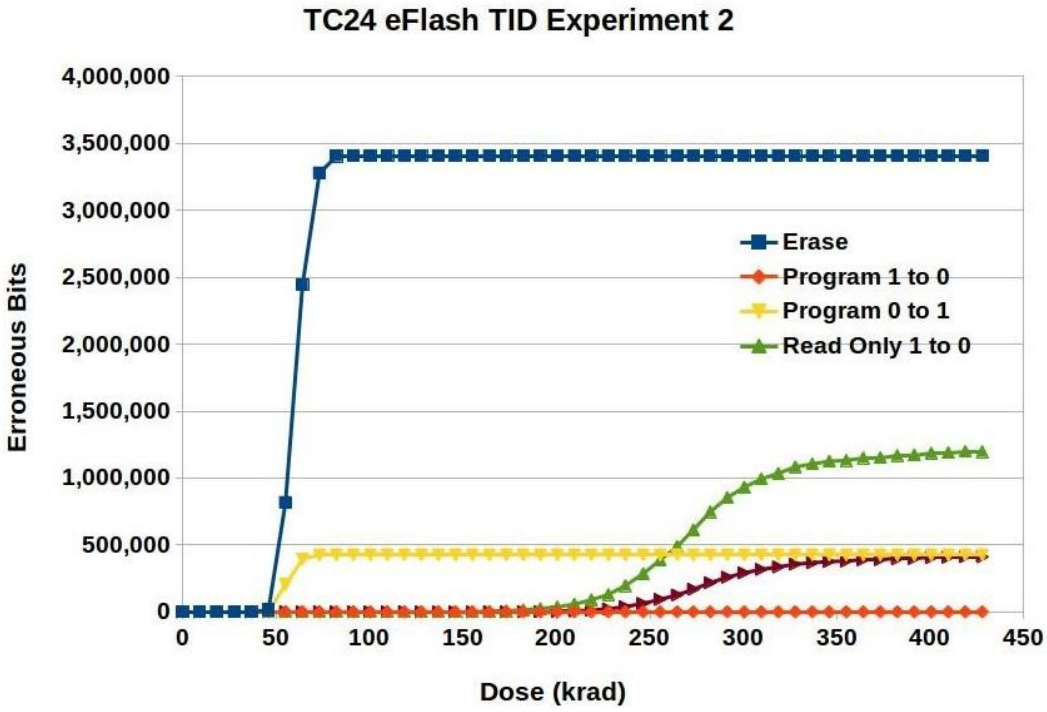


Fig. 2.19 Bit errors observed in the eFlash TID experiment 2.

One million bits failed in the erase operation at 60 krad dose (Fig. 2.19) for experiment 2. Half a million bits failed at 70 krad dose in the program 0 to 1 operation for experiment 2. One million bits failed in the erase operation at 36 krad dose for experiment 1. Close to one million bits failed at 200 krad dose in the program 0 to 1 operation for experiment 1. The erase operation failed early than other operations for both experiment 1 and experiment 2, rendering erase mostly non-functional. Increased voltage may increase the survival dose of the chip.

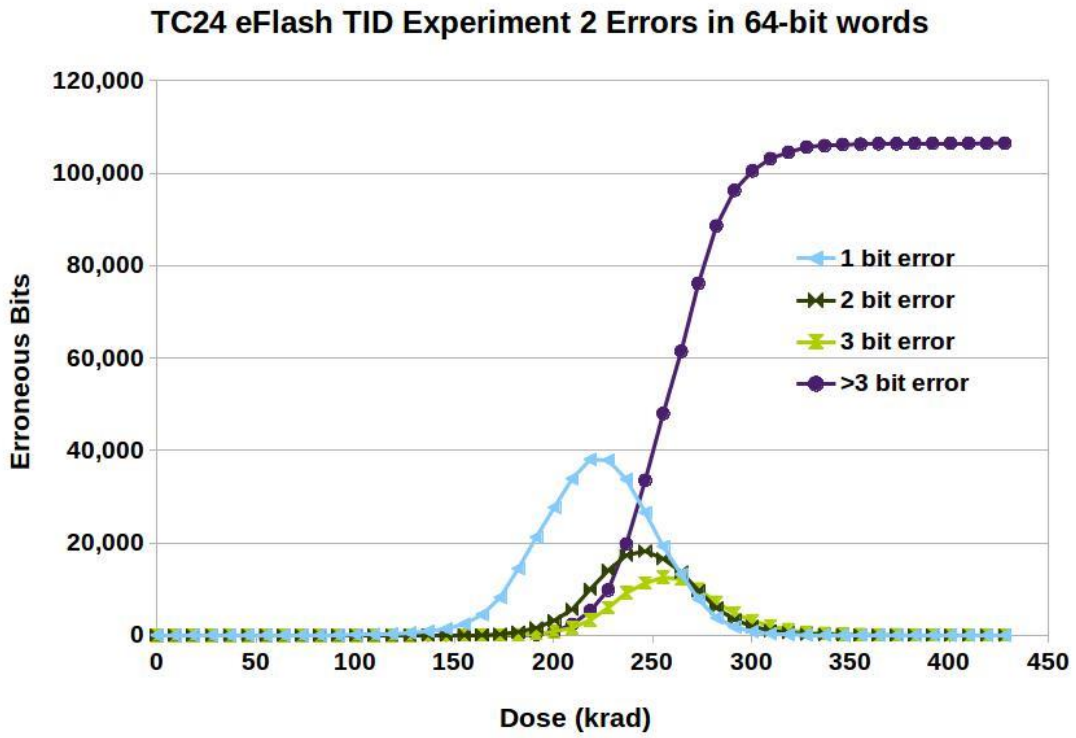


Fig. 2.20 Word errors observed in the eFlash TID experiment 2.

Each address location holds 64-bit words (32-bit per bank). If a 3-bit error detection and correction (EDAC) logic is incorporated, then up to 3-bit errors can be corrected. As shown in Fig. 2.20, with a 3-bit EDAC, the flash memory may survive until 200 krad dose.

For experiment 1 (Fig. 2.21), the current drawn by the charge pump increased with an increase in the TID radiation dose. However, for experiment 2, the current drawn by the charge pump was constant with an increase in the TID radiation dose. Even though, the eFlash memory failed in a similar fashion for erase, program and read operations for both experiment 1 and experiment 2, the only main difference observed in the results for experiment 1 and experiment 2 is the charge pump current. The exact reason for the current remaining constant in experiment 2 is unknown. It is highly possible that incorrect connections were made for current measurement in experiment 2.

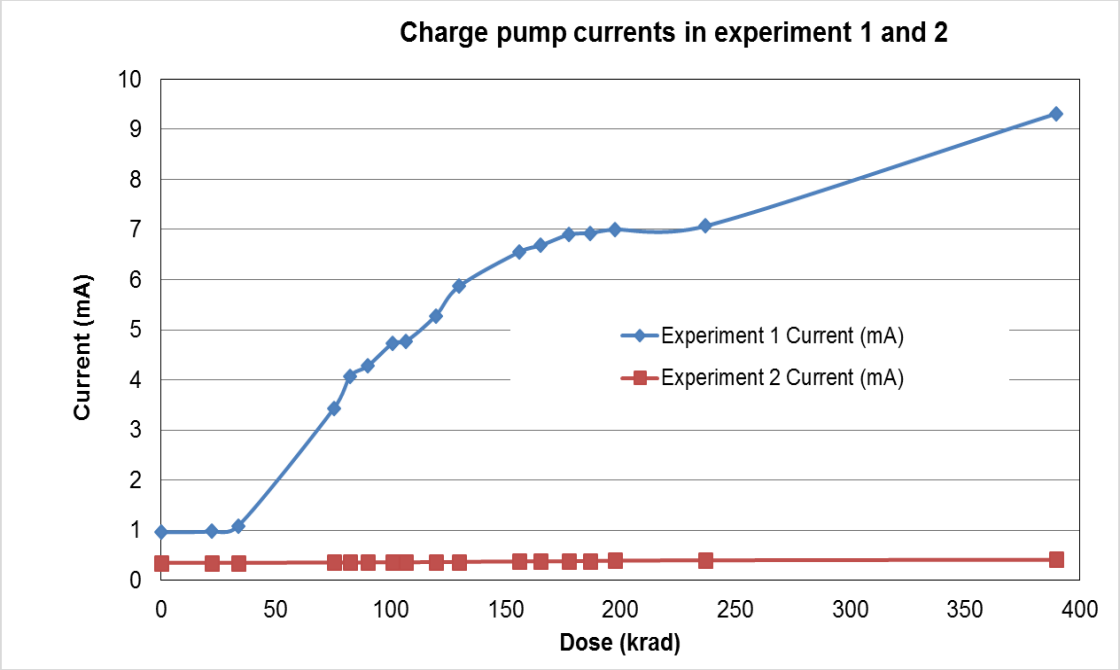


Fig. 2.21 Charge pump currents in the eFlash TID experiments 1 and 2 vs. dose.

CHAPTER 3. HERMES VALIDATION

3.1. HERMES Architectural Overview

The HERMES is a 32-bit MIPS 4Kc core based microprocessor. It incorporates a combination of architectural, micro-architectural and circuit level techniques to provide radiation hardness [Gogula15]. It consists of dual mode redundant (DMR) speculative pipeline operation and triple mode redundant (TMR) architectural state except for memories [Hind11]. An operating system such as Linux can run on the HERMES core since it includes full MMU support. The HERMES is capable of soft-error recovery, allowing programmer control of the recovery process and error reports. It has special instructions that allow silicon validation of soft-error recovery without broad beam testing. This capability is useful to have a test program that can be fully pre-validated on the silicon before exposing hardened processors to radiation.

A high-level block diagram of the HERMES processor without radiation hardening features is shown in Fig. 3.1, and it has 11 functional units. The bus interface unit (BIU) is the interface between the processor core and the external bus. It receives instruction fetch requests, and loads and store requests from the instruction fetch unit (IFU) and the data cache unit (DCU) respectively. It includes a write buffer for buffering store requests from the DCU before sending them out onto the external bus. The IFU is responsible for delivering instructions to the core pipeline. It contains a 16 KB, 4-way set associative instruction cache (I-cache), one fill buffer, and a micro instruction translation look-aside buffer (ITLB) for translating virtual addresses to physical addresses. The instruction decode unit (IDU) is responsible for decoding instructions from the IFU. The register file unit (RFU) contains the general-purpose register file with 32 32-bit registers. The

instruction execution unit (IEU) is responsible for executing all the instructions except for multiply and divide instructions. The multiply/divide unit (MDU) is responsible for executing all multiply and divide instructions. The DCU is responsible for servicing load and store instructions. It contains a 16 KB, 4-way set associative, write-through, a read-allocate only data cache (D-cache), one fill buffer, and a micro data TLB (DTLB) for translating virtual addresses to physical addresses. The memory management unit (MMU) contains the joint TLB (JTLB), which is a 16 dual entry TLB providing virtual address to physical address translations for the ITLB and DTLB. The coprocessor 0 unit (C0U) contains all of the coprocessor 0 (system coprocessor) registers. The JTAG unit (JTU) contains the joint test action group (JTAG) debug and testability logic. The clock management unit (CMU) provides the entire clock and power management functionality.

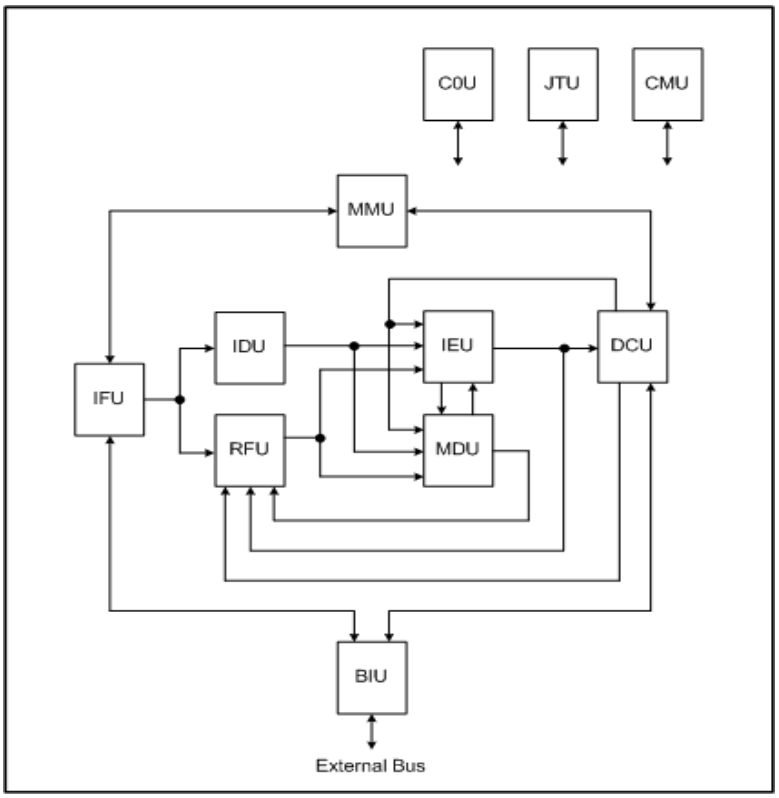


Fig. 3.1 High-level block diagram of the HERMES processor.

3.2. HERMES Test Bench Setup

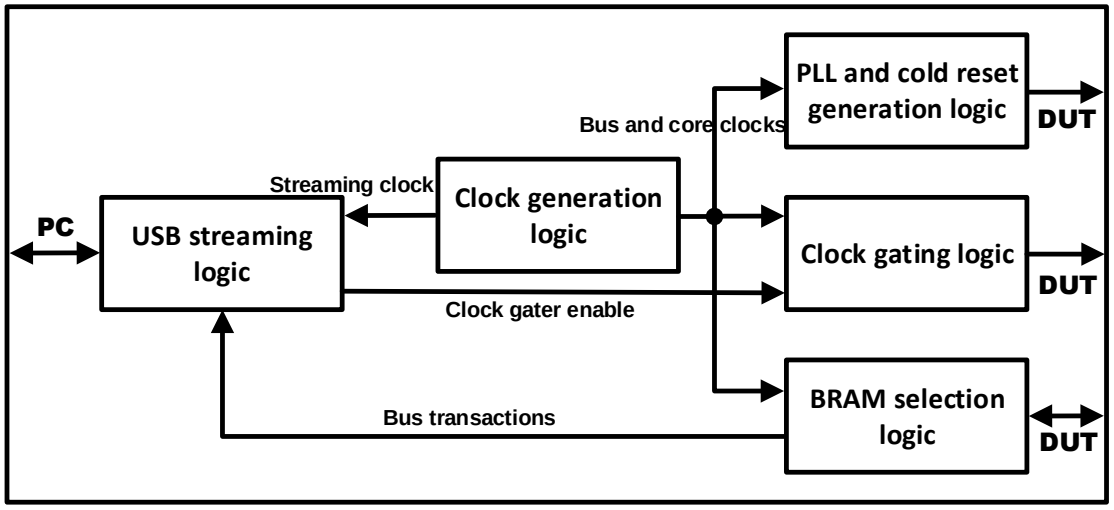


Fig. 3.2 High-level block diagram of the HERMES test bench.

Fig. 3.2 shows a high-level block diagram of all the five different modules present in the HERMES test bench (TB). This TB is mapped inside the Spartan 3 XC3S4000 FPGA. The FPGA supports 96 block random access memories (BRAMs) and four digital clock managers (DCMs). Each BRAM has 512 36-bit locations; each 36-bit location has 32 data bits and 4 parity bits. Each DCM can generate from 18 MHz to 210 MHz clock frequencies. The USB streaming logic (Fig. 3.2) streams a record of the bus transactions to the PC. The clock generation logic uses the DCMs to generate core, bus, and streaming clocks. The PLL and cold reset generation logic gives the required initial reset sequence to the HERMES processor (DUT) to synchronize the bus and internal core clocks. The clock gating logic stops the clocks to the HERMES processor when the USB streaming logic is not ready. The BRAM selection logic gives the read data output or stores the write data input based on the address input provided by the HERMES. It acts as a memory hierarchy of the (simulated) system surrounding the HERMES.

3.2.1. USB Streaming and Clock Generation Logic

The USB streaming logic uses four BRAM's (Fig. 3.3); each BRAM holds control, address, read data and write data bits respectively. The write interface of each BRAM operates with the bus clock (CLK_BUS) and records all the HERMES bus transactions. When the BRAMs are full (IO_BRAM_Full shown in Fig. 3.4 is logic '1'), both the test bench and the HERMES clocks are gated off to avoid future transactions. The read interface of each BRAM works on the USB streaming clock (always 48 MHz) and sends the bus transactions to the USB to PC stream on a first in first out (FIFO) basis every clock cycle. If all the BRAM's are empty or the USB is not ready to receive data (DataOutBusy is logic '1'), then valid data is not sent (DataOutWE is logic '0') to the USB streaming FIFO.

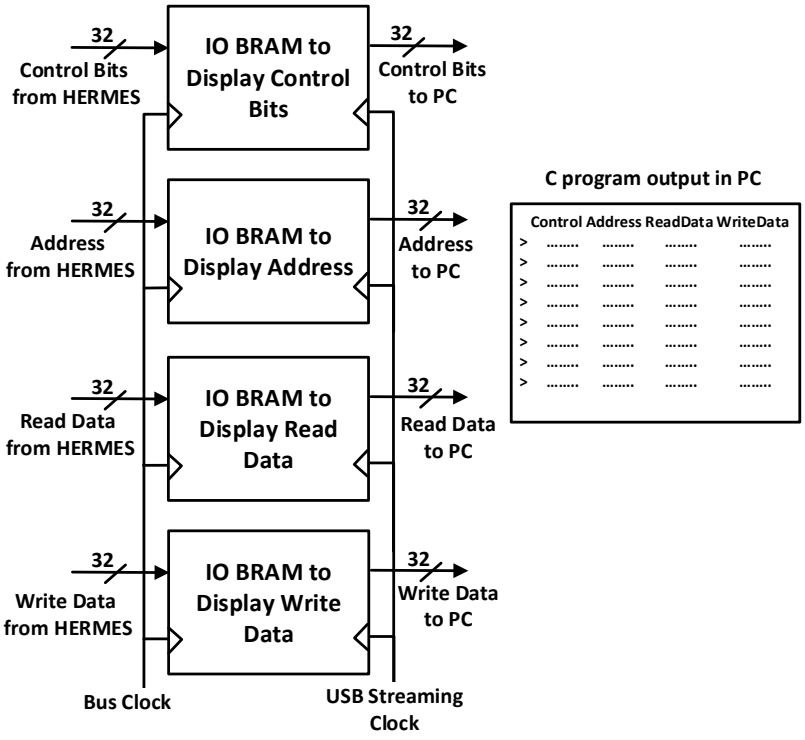


Fig. 3.3 High-level block diagram of the USB streaming logic.

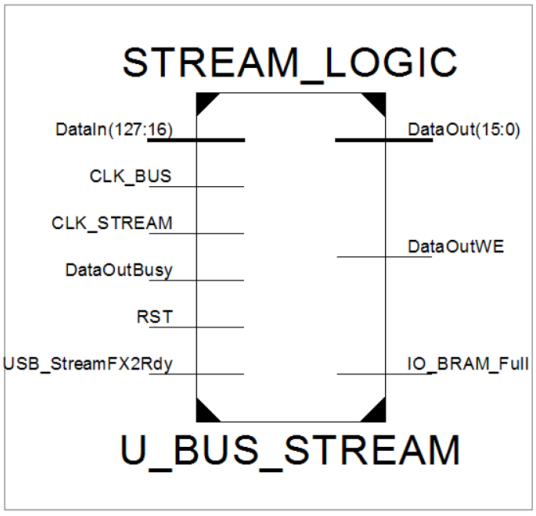


Fig. 3.4 Symbol view of the USB streaming logic.

The ZestSC2board has only one 48 MHz crystal oscillator onboard. Two DCMs are used for the HERMES testing, one for the USB streaming and the other one for the regular user clock. The USB streaming DCM generates the required 48 MHz clock (shown in Fig. 3.5) for the streaming interface from the crystal oscillator. The regular user DCM uses the same 48 MHz crystal clock as the input, and can generate any frequency from 18 MHz to 210 MHz using the digital frequency synthesizer (DFS). The DFS is used even though it has higher jitter compared to the delay locked loop (DLL) because it provides a wide and flexible range of output frequencies than the DLL.

The higher clock frequencies for HERMES core are generated inside the TC24 by using phase-shifted clocks (refer to section 1.1.5). The phase-shifted clocks are generated using the cascaded DCM. For example, to generate multiplied-by-2 core clock frequency XOR_CLK0 and XOR_CLK90 (90 degrees phase-shifted) are needed. The CLK0 output signal from the cascaded DCM is used as the XOR_CLK0 signal because it is in phase with the input clock to the DCM. The CLK90 output signal from the same cascaded DCM

is used as the XOR_CLK90 signal because it is a 90 degrees phase-shifted version of the input clock to the DCM. The CLKDV output signal from the same cascaded DCM is used as the bus clock (CLK_BUS). The divider value used to generate the CLKDV signal is set to the bus ratio of the HERMES.

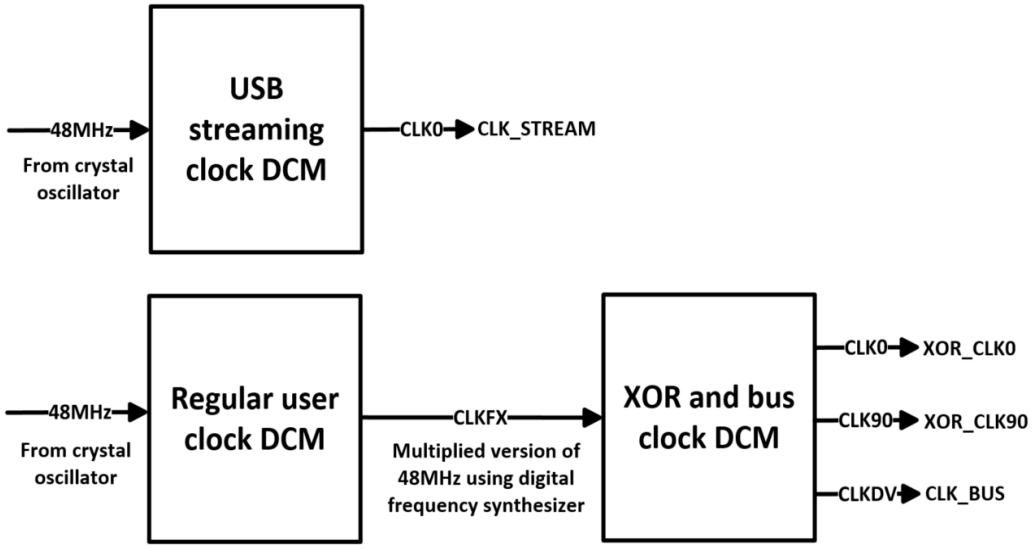


Fig. 3.5 Block diagram of the clock generation logic.

3.2.2. PLL and Cold Reset Generation Logic

The reset logic (Fig. 3.6) generates the SI_PLL_Reset and the SI_ColdReset reset sequences to the HERMES during the initial few clock cycles. The SI_PLL_Reset is a reset signal for the PLL. This signal is also used to load the PLL configuration and bus-to-core clock ratio registers. The SI_ColdReset is a hard reset signal and causes a reset exception in the core. As mentioned, this signal is required for synchronizing the internal processor core clock with the external bus clock.

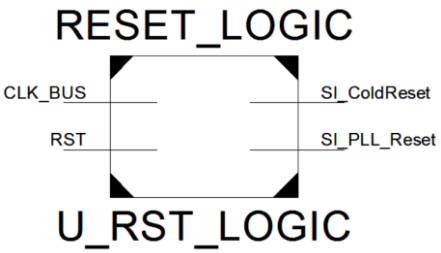


Fig. 3.6 Symbol view of the reset generation logic.

Initially the SI_PLL_Reset is de-asserted (logic '0') and the SI_ColdReset is asserted (logic '1'). On the first negative edge of the bus clock (Fig. 3.7), the SI_PLL_Reset is pulled high, and the SI_ColdReset continues in the logic high. The SI_PLL_Reset is de-asserted (logic low) on the third negative edge of the bus clock while the SI_ColdReset remains asserted (logic high). The SI_ColdReset is held high until the clocks from the PLL become stable. Hence, the SI_ColdReset is pulled low on the seventh negative edge of the bus clock.

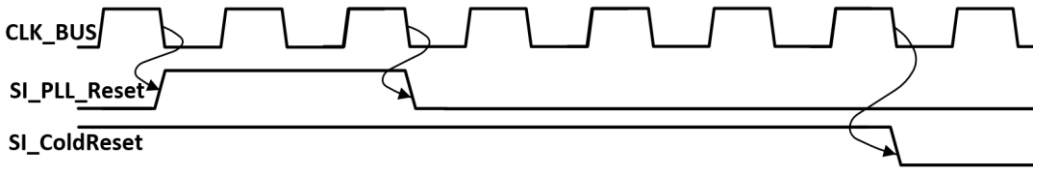


Fig. 3.7 Timing diagram for the PLL and cold reset signals.

3.2.3. Clock Gating Logic

When the USB streaming interface BRAMs are full (i.e., not ready), new bus transactions cannot be recorded (refer to section 3.2.1). Hence, in this condition the bus and XOR clocks (Fig. 3.8) are gated so that the HERMES will not issue new bus transactions.

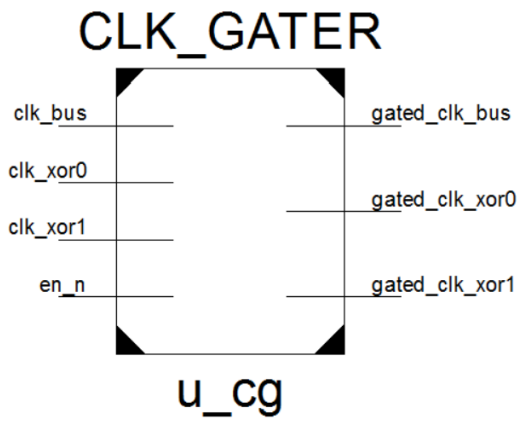


Fig. 3.8 Symbol view of the clock gating logic.

Since clocks cannot be gated to the HERMES using a standard AND gate, a clock gater module (made using a negative latch and an AND gate) is used. The clock gater module stops both the bus clock (CLK_BUS) and the core clocks (XOR_CLK0 and XOR_CLK90) when the USB streaming interface is not ready. The clocks are subsequently released when the USB streaming interface is ready. Thus, filling the streaming interface stalls testing.

3.2.4. BRAM Selection Logic

The address bus (Fig. 3.9) is a direct input to the BRAMs. EB_AValid, EB_Write, ADDRESS and SI_ColdReset signals determine the 2-bit select signals, which are flopped on the clock edge. The flopped select signals select the EB_RData from either one of the BRAMs, or constant values. There are four possible cases to determine the value of the EB_RData. First, when the cold reset is asserted the EB_RData output will have 22nd to 26th bits equal to the bus ratio, and the remaining bits are zeros. Second, when there is an invalid read transaction all 32-bit values of the EB_RData are zeros. Third, when there is an instruction fetch (EB_Instr is asserted) and the address points to the 0000:0000 BRAM,

the data output of this BRAM is assigned to EB_RData. Finally, when there is an instruction fetch and the address points to the 1FC0:0000 BRAM, the data output of this BRAM is assigned to EB_RData. When there is a valid read transaction, EB_RdVal signal is asserted on the negative edge of the next bus clock cycle. Depending on the test case, multiple BRAMs can be included to provide all the memory needed for that test.

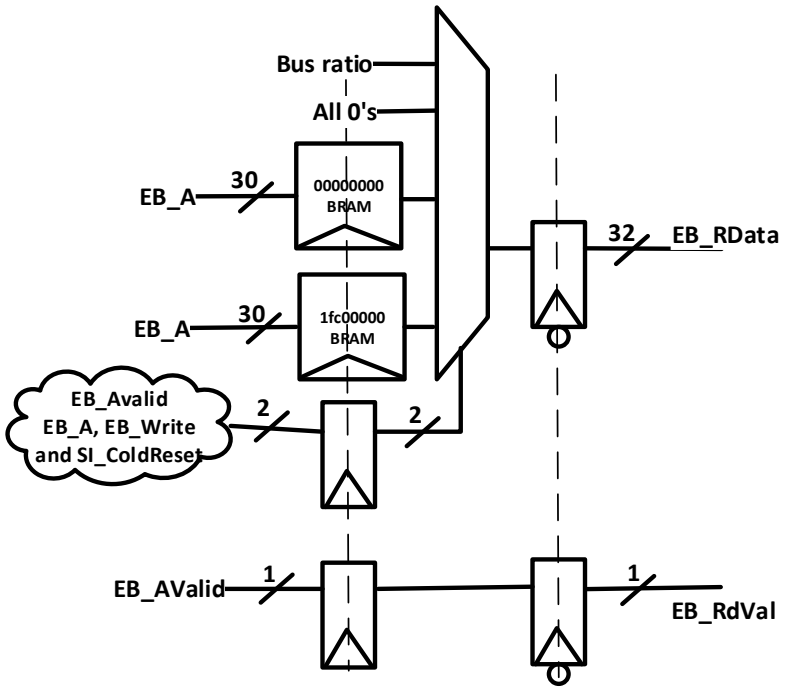


Fig. 3.9 Block diagram of the BRAM selection logic.

3.3. Change in Test Bench Sampling Edge from Negative to Positive

3.3.1. Bus Failing on the Negative Edge Case

The initial FPGA test bench had a number of timing issues. As shown in Fig. 3.10, the bus clock is generated inside the test bench (FPGA) and goes to the HERMES (DUT) through the PCB. The HERMES outputs the address on the positive edge of the bus clock, and the address goes to the test bench through the PCB. Since the clock is generated from

the test bench, the test bench may see the negative edge of the clock before the address is sampled (because of the PCB delay). At higher frequencies (greater than 40 MHz), the BRAMs and flip-flops will miss their setup times for the address signals. Hence, valid data output is sent on the next negative edge of the bus clock instead of the same negative edge of the bus clock. The BRAMs inside the test bench have large setup time compared to a flip-flop. For medium frequencies, less than 40 MHz and greater than 20 MHz, the BRAMs miss their setup times and flip-flops work properly. In this case, the HERMES receives read valid signal (EB_RdVal) indicating the read data as valid even though the data are invalid, causing a failure due to the test board timing.

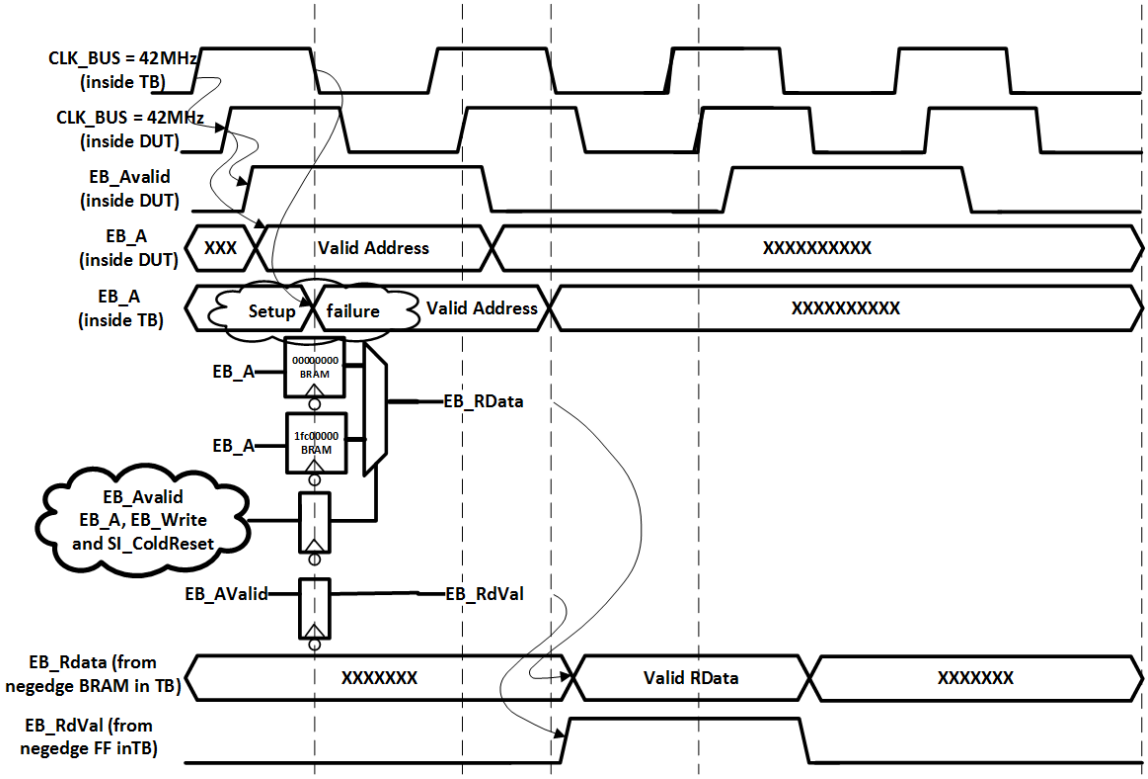


Fig. 3.10 Timing problem in the negative edge sampling case.

As shown in the trace output (Fig. 3.11), the HERMES output (EB_AValid) is not sampled properly by the BRAM on the same negative edge of the bus clock (SI_ClkIn) in

the test bench. Instead, the read data (EBRData[28]_chip) output was given on the next negative edge of the bus clock.

Fig. 3.11 Trace output of the bus failing case.

3.3.2. Timing Fix on the Positive Edge Case

As shown in Fig. 3.12, the BRAM samples the address on the positive edge of the bus clock, one clock cycle after the processor issues the valid address transaction. The BRAM then sends the valid read data to the negative edge triggered flip-flops. The negative edge triggered flip-flops then send the valid read data to the processor. The negative edge triggered flip-flops are added to avoid setup time failures in the processor. For bus clock frequencies less than 42 MHz, the processor receives the valid read data after one clock cycle. Now, with the bus clock frequency running at 42 MHz, and the bus ratio value as eight, the processor core clock can run at 336 MHz (42 * 8).

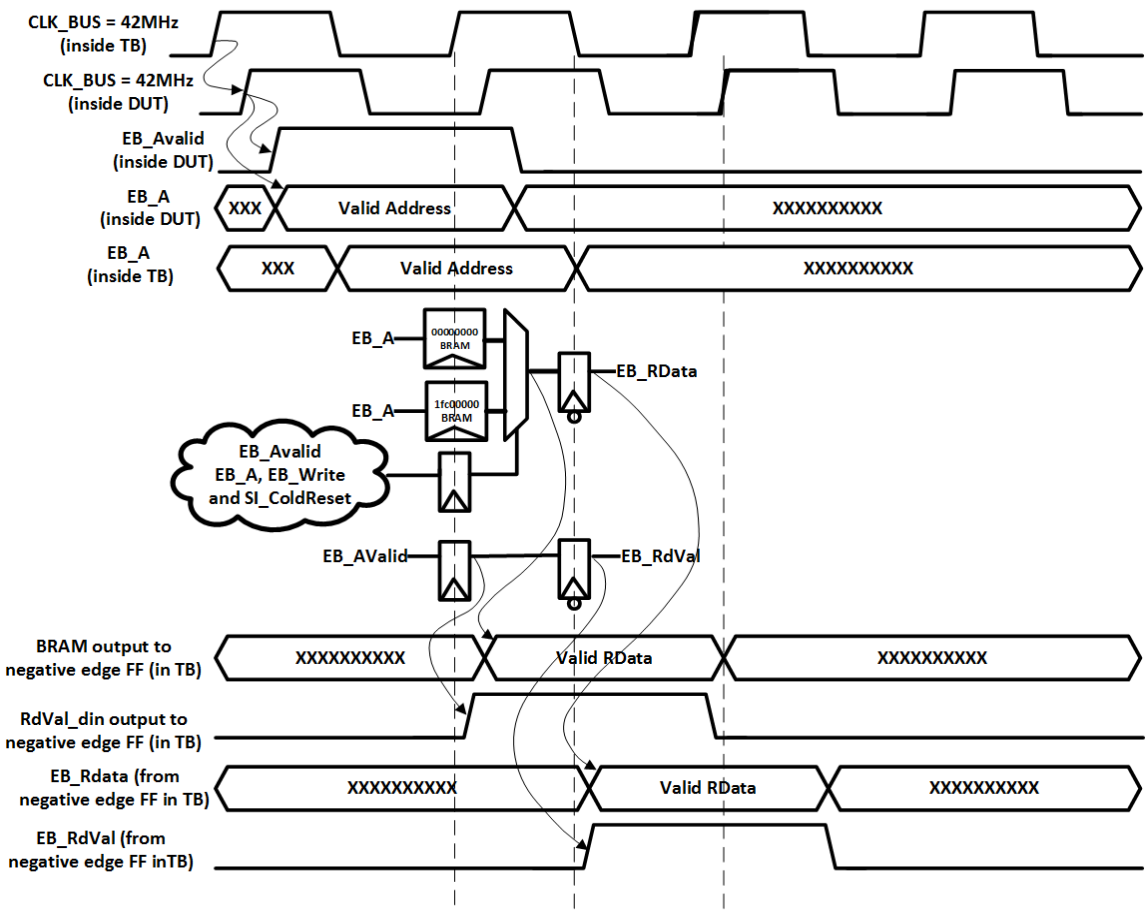


Fig. 3.12 Timing fix in the positive edge sampling test bench.

3.4. HERMES Debug Results

Part 13 was used for all the HERMES tests since it works at the lowest voltage (0.65 V) compared to the others parts. However, the lower voltage is limited by the I/O level shifters and has nothing to do with the HERMES core behavior [Gogula15]. The HERMES core should be functional even at the subthreshold voltages, based on simulation and analysis of the circuits.

3.4.1. Tests with Caches Activated

The "Hello world" test case was designed to determine that the processor would run the code from the cached section of memory successfully. To run the program in the

cached space, the cache needs to be initialized, and the initialization instructions are highlighted in the red box in Fig. 3.13. These instructions move the configuration bits to a general-purpose register, change the cache enabler bits to turn the cache on, and then move the configuration bits back. The test case also initializes all registers in the register file (RF) and displays “Hello World!” on the bus in an infinite loop. After the instructions are executed once, only the store words will be on the bus, as everything else is cached.

```
132  bfc00460:   0000c821    move    t9,zero
133  bfc00464:   0000d021    move    k0,zero
134  bfc00468:   0000d821    move    k1,zero
135  bfc0046c:   0000e021    move    gp,zero
136  bfc00470:   0000e821    move    sp,zero
137  bfc00474:   0000f021    move    s8,zero
138  bfc00478:   40048000    mfc0    a0,c0_config
139      ...
140  bfc00488:   34840003    ori a0,a0,0x3
141  bfc0048c:   40848000    mtc0    a0,c0_config
142      ...
143
144  bfc0049c <oloop>:
145  bfc0049c:   3c09bfc0    lui t1,0xbfc0
146  bfc004a0:   252904f8    addiu   t1,t1,1272
147  bfc004a4:   3c0abfc0    lui t2,0xbfc0
```

Fig. 3.13 Cached hello world required instructions.

The HERMES is able to run the code from the cached section successfully because only the data is stored onto the bus (Fig. 3.14), as all the instructions are cached.

```
16'b0000111110000101    1 000c0000 00000000 1 6c6c6548 Hell
16'b0000111110000101    1 000c0004 00000000 1 6f77206f o.wo
16'b0000111110000101    1 000c0008 00000000 1 0a646c72 rld.
16'b0000111110000101    1 000c0000 00000000 1 6c6c6548 Hell
16'b0000111110000101    1 000c0004 00000000 1 6f77206f o.wo
16'b0000111110000101    1 000c0008 00000000 1 0a646c72 rld.
16'b0000111110000101    1 000c0000 00000000 1 6c6c6548 Hell
16'b0000111110000101    1 000c0004 00000000 1 6f77206f o.wo
16'b0000111110000101    1 000c0008 00000000 1 0a646c72 rld.
```

Fig. 3.14 Cached hello world output.

3.4.2. Testing the HERMES Data Caches

There are three steps in the test case designed to verify the data cache functionality. The first step is to write the data to the cacheable memory locations. This writes the BRAM, but not the data cache since the HERMES uses a non-write allocate data cache policy—this keeps transient data from sweeping the cache clean. The second step is to read the data, modify it, and write the data back to the memory locations. The read allocates the data cache locations. The third step is to modify the data to ensure the data are changing when the processor is reading from the cache. The write appears on the bus and the cache (as it is a write-through cache), and the write is apparent on the bus via the streaming interface. Now repeat the second step. Since, the values are cached the processor reads from its cache resulting in no bus activity. However, writes are observed on the bus. After the data are modified again, the new values appear on the bus. This validates that correct data were read previously from the cache (also confirming the writes). Fig. 3.15 shows the pseudo code for the test.

```
loop1: for $t1 0x9000.0000 to 0x9000.0024 {
        increment $t2 by 1. $t2 is the data that will be stored (1 first cycle, 2 next cycle, ...,
        0xA last cycle)
        store the data from $t2 into BRAM address $t1
        increment $t1 by 4, point to next address for next cycle
}
load $t1 with 0x9000.0000 to reset this address pointer.

loop2:for $t1 0x9000.0000 to 0x9000.0024 {
        $t2 = read data from BRAM address $t1
        increment $t2 by 16
        store changed data of $t2 into BRAM address $t1
        increment $t1 by 4 (address pointer)
}

pre_loop3:
load $t1 with 0x9000.0000 to reset this address pointer

loop3:
for $t1 0x9000.0000 to 0x9000.0024 {
        $t2 = read data from BRAM address $t1
        increment $t2 by 16
        store changed data of $t2 into BRAM address $t1
        increment $t1 by 4 (address pointer)
}
jump to pre_loop3 (reset $t1)
```

Fig. 3.15 Pseudo code for testing the data cache.

Fig. 3.16 shows the write data are not only stored into the BRAM but also stored into the cacheable addresses (because EB_Write is 1). Hence, the data cache is initialized properly.

```
0 : CardID = 0x00000001, SerialNum = 0x0000056f, FPGAType = 2
    Control       Avalid Address   RData EB_Write WData
  EEEEEEEESSSSSEEE
  BBBBBBBBIIIIIBBB
  bbbWBBBBEERSTAiw
  uflWEEEERXPLiVnr
  riaB3210LL Emasi
  srsE    Eeltt
  tst    Prire
   t     Id
        n
        t

16'b0000111110000101    1 10000024 00000000 1 0000002a ....
16'b0000111110000001    0 10000024 00000000 1 0000002a ....
16'b0000111110000001    0 10000024 00000000 1 0000002a ....
16'b0000111110000001    0 10000024 00000000 1 0000002a ....
16'b0000111110000001    0 10000024 00000000 1 0000002a ....
16'b0000111110000001    0 10000024 00000000 1 0000002a ....
16'b0000111110000001    0 10000024 00000000 1 0000002a ....
16'b0000111110000001    0 10000024 00000000 1 0000002a ....
16'b0000111110000001    0 10000024 00000000 1 0000002a ....
16'b0000111110000001    0 10000024 00000000 1 0000002a ....
16'b0000111110000001    0 10000024 00000000 1 0000002a ....
16'b0000111110000001    0 10000024 00000000 1 0000002a ....
16'b0000111110000101    1 10000000 00000000 1 00000031 ....
16'b0000111110000001    0 10000000 00000000 1 00000031 ....
16'b0000111110000001    0 10000000 00000000 1 00000031 ....
16'b0000111110000001    0 10000000 00000000 1 00000031 ....
16'b0000111110000001    0 10000000 00000000 1 00000031 ....
16'b0000111110000001    0 10000000 00000000 1 00000031 ....
16'b0000111110000001    0 10000000 00000000 1 00000031 ....
16'b0000111110000001    0 10000000 00000000 1 00000031 ....
16'b0000111110000001    0 10000000 00000000 1 00000031 ....
                              … (infinite loop)
```

Fig. 3.16 Correct result for the data cache test.

3.4.3. Testing the HERMES TLBs

This test verifies the TLB functionality on the HERMES. It uses a TLB mapping and then writes values to the registers. The correctness of TLB mapping was verified by running the same code on an MIPS simulator, VMIPS.

To execute this test, subroutines used are Setup, Mumble, and DumpRF.

1. Setup Pseudocode:

    - Clear out all registers -> set them to 0.
    - Turn on cache (mfc0 $4, $16 instructions)
    - Clear out the TLBs
    - Initialize TLBs
    - Call Mumble
    - Call DumpRF
    - Display Test End

2. Mumble Pseudocode:

    - Place the numerical value of the register in itself. $0 = 0, $1 = 1, etc.
    - Return to setup code.

3. DumpRF Pseudocode:

    - Store stackpointer = 0xa00c.02a0
    - Store ra into 0xa00c.02ac
    - Set up outgoing register address to 0xa00c.0070
    - Store words on the bus to indicate it is a dump
    - Dump the registers one by one with store words. $0, $1, etc.
    - Load stackpointer
    - Load ra
    - Jump back to setup code.

The RF Dump operation is shown in Fig. 3.17. The output shows that the values were written properly during mumble. I-cache is also working, because the only transactions seen on the bus are writes after all instructions are cached. In addition, when compared to the clock counter in the non-cached test, the difference in clocks shows that the I-cache is working. Because this output (Fig. 3.17 and Fig. 3.18) matches the output given from VMIPS, this test passes.

```
0 : CardID = 0x00000001, SerialNum = 0x0000056f, FPGAType = 2
clock  EEEEEEEESSSSSEEE    E Address    RData  E WData
clock  BBBWBBBBIIIIIBBB    B Address    RData  B WData
clock  ____________________ _ Address    RData  _ WData
clock  BBBWBBBBEERSTAIW    A Address    RData  W WData
clock  uFLWEEEERXPLiVnr    V Address    RData  r WData
clock  riaB3210LL Emasi    a Address    RData  i WData
clock  srsE       Eeltt    l Address    RData  t WData
clock  tst        Prire    i Address    RData  e WData
clock  t            Id     d Address    RData    WData
clock                n       Address    RData    WData
clock                t       Address    RData    WData

00600  0000111111000101    1 000c0070 00000000 1 44204652 RF.D
00601  0000111111000100    1 000c0294 3a706d75 0 44204652
00609  0000111111000101    1 000c0074 00000000 1 3a706d75 ump:
00610  0000111111000100    1 000c0298 0000000a 0 3a706d75
00623  0000111111000101    1 000c0078 00000000 1 0000000a ....
00627  0000111111000101    1 000c00e0 00000000 1 00000000 ....
00631  0000111111000101    1 000c00e4 00000000 1 00000001 ....
00635  0000111111000101    1 000c00e8 00000000 1 00000002 ....
00639  0000111111000101    1 000c00ec 00000000 1 00000003 ....
00643  0000111111000101    1 000c00f0 00000000 1 00000004 ....
00647  0000111111000101    1 000c00f4 00000000 1 00000005 ....
00651  0000111111000101    1 000c00f8 00000000 1 00000006 ....
00655  0000111111000101    1 000c00fc 00000000 1 00000007 ....
00659  0000111111000101    1 000c0100 00000000 1 00000008 ....
00663  0000111111000101    1 000c0104 00000000 1 00000009 ....
00667  0000111111000101    1 000c0108 00000000 1 0000000a ....
00671  0000111111000101    1 000c010c 00000000 1 0000000b ....
00675  0000111111000101    1 000c0110 00000000 1 0000000c ....
00679  0000111111000101    1 000c0114 00000000 1 0000000d ....
00683  0000111111000101    1 000c0118 00000000 1 0000000e ....
00687  0000111111000101    1 000c011c 00000000 1 0000000f ....
00691  0000111111000101    1 000c0120 00000000 1 00000010 ....
```

Fig. 3.17 RF dump output.

Once again, “Test end” is displayed as output on the bus (Fig. 3.18) after the test.

This shows that the RF dump returned to setup as expected; therefore, the test passes.

```
00762  0000111111000101    1 000c0154 00000000 1 00000000 ....
00766  0000111111000101    1 000c0158 00000000 1 a00c02a8 ....
00771  0000111111000101    1 000c015c 00000000 1 00000000 ....
00772  0000111111000100    1 000c02ac 800000f8 0 00000000
00786  0000111111000100    1 000c0060 74736554 0 00000000
00794  0000111111000101    1 000c0000 00000000 1 74736554 Test
00795  0000111111000100    1 000c0064 646e6520 0 74736554
00803  0000111111000101    1 000c0004 00000000 1 646e6520 .end
```

Fig. 3.18 Output displaying test end message on the bus.

3.4.4. Power Determination

Multiply-accumulate (MAC) instructions give a good estimate for power dissipation because they exercise the multipliers (the most power hungry block). Table 3-1 summarizes the power dissipation as a function of the core frequency. Power was calculated from the supply voltage and the DUT supply current was measured via a series ammeter. The test outputs were minimized to ensure that there were no breaks due to the streaming delays, but sufficient to guarantee correct functionality.

Core Frequency (MHz)	Bus Frequency (MHz)	Voltage (V)	Current (A)	Power (W)
96	48	1.2	0.042	0.0504
192	48	1.2	0.075	0.0900
288	48	1.2	0.109	0.1308
312	52	1.2	0.119	0.1428

Table 3-1 Power vs core frequency for MAC instructions.

From Table 3-1 and Fig. 3.19, the maximum power dissipation at 312 MHz core frequency is 142.8 mW. Linearity of the power dissipation vs. core frequency (Fig. 3.19) is as expected.

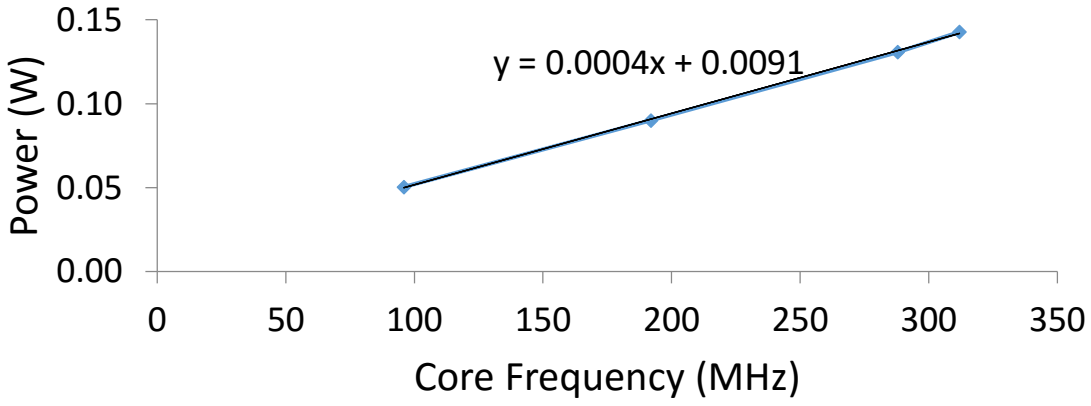


Fig. 3.19 Power vs core frequency for MAC instructions.

CHAPTER 4. SUMMARY

All the key components of the HERMES are confirmed as functional. These include the data cache, instruction cache, and TLBs. In addition, many instructions in the MIPS 4Kc instruction set have been tested for functionality. All instructions except for one SEE checker are confirmed functional. There is a critical timing issue in the RF write back checker. The fix appears to be straightforward, and it will be fixed in the future version of the HERMES. The next version of the HERMES will be manufactured at 65-nm ultra-low power (sub 400 mV VDD) process. HERMES has no uncorrectable soft errors in 500+ events with proton testing to over 10^{11} protons/cm^2 fluence.

The HERMES prototype silicon works with a minimum supply voltage of 1.2 V at a core frequency of 312 MHz. When the supply voltage is increased to 1.4 V its maximum frequency of operation increases to 336 MHz [Gogula15]. The HERMES can operate with VDD as low as 650 mV (limited by foundry I/O's). With a MAC intensive program running from the cache, the power dissipation is measured as 143 mW at a core frequency of 312 MHz. This places the HERMES core among the highest frequency, lowest voltage and lowest power radiation-hardened processors published. Since producing correct phase alignment for the XOR clock multiplier is difficult, the speed measurements are pessimistic.

The requirement for the eFlash prototype is to demonstrate the key functionality necessary to configure a Xilinx FPGA (compatible with a Platform Flash in-system programmable configuration PROM) while providing an appropriate level of radiation hardness. However, the embedded flash chip failed to provide the required amount of radiation hardness. In eFlash TID experiment 2, the read-only operation started failing after

150 krad TID dose and the erase operation started failing after 50 krad dose, rendering erase mostly non-functional. If we incorporate 3-bit EDAC, then the embedded flash memory may survive until 200 krad TID dose.

The experimental results on the standalone version of the same flash memory [Clark15] show that the read-only operation exhibited no failures in testing up to 300 krad TID dose. The erase operation failures were observed before 100 krad dose, and thus the erase operation is dominating the radiation hardness in the standalone version too. Even though the same flash memory is used in both the standalone and embedded versions, the embedded version results are disappointing.

The test structures consist of five shift registers, each 1024 bits long. Three shift registers make use of temporally radiation-hardened flip-flops while the remaining two shift registers make use of conventional unhardened flip-flops. These shift registers are used as a reference while beam testing is conducted to see how many errors are observed in the structures that are not radiation hardened vs. how many errors are observed in the structures that are radiation hardened. Three errors were observed in the conventional unhardened flip-flops, and no errors were observed in the temporally radiation-hardened flip-flops. Very few errors were observed in the conventional unhardened flip-flops because the test structures area was too small to get good statistics in the time that was allotted at the beam.

REFERENCES

[IEEE01] IEEE Std 1149.1-2001, “IEEE Standard Test Access Port and Boundary-Scan Architecture,” IEEE, USA, 2001.

[IEEE02] IEEE Std 1532-2002, “IEEE Standard for In-System Configuration of Programmable Devices,” IEEE, USA, 2002.

[Naseer08] R. Naseer and J. Draper, “DEC ECC design to improve memory reliability in sub-100nm technologies,” *Proc. 15th IEEE Int. Conf. Electron. Circuits Syst. ICECS 2008*, pp. 586–589, 2008.

[Xilinx09] Xilinx, “Platform Flash PROM”, User Guide, UG161, Oct. 2009.

[Xilinx10] Xilinx, “Platform Flash In-System Programmable Configuration PROMs”, Product Specification, DS123, May. 2010.

[Xilinx12] Xilinx, “Virtex-5 FPGA Configuration”, User Guide, UG191, Oct. 2012.

[Micro12] Microchip, “Preliminary Target Specification (MRD)”, Pfm_390t105kx32_v1 ESF3-90, Jul. 2012.

[Patt13] D. W. Patterson, L. T. Clark, “TC23 Microarchitecture Specification”, v3.0, May. 2013.

[Orange10] Orange Tree Technologies, “ZestSC2 User Guide”, v1.1, Mar. 2010

[Mentor04] Mentor Graphics, “ModelSim: Advanced Verification and Debugging”, v 6.0b, Nov. 2004

[Carlo09] S. Di Carlo, P. Prinetto, A. Scionti, J. Figueras, S. Manich, and R. Rodriguez-Montañés, “A low-cost FPGA-based test and diagnosis architecture for SRAMs,” *1st Int. Conf. Adv. Syst. Test. Valid. Lifecycle, VALID 2009*, pp. 141–146, 2009.

[Gujja15] A. Gujja, “Redundant Skewed Clocking of Pulse-Clocked Latches for Low Power Soft-Error”, Master’s thesis, Arizona State University, 2015.

[Clark15] L. T. Clark, S. Member, K. E. Holbert, S. Member, J. W. Adams, H. Navale, and B. C. Anderson, “Evaluation of 1 . 5-T Cell Flash Memory Total Ionizing Dose Response,” vol. 62, no. 6, pp. 2431–2439, 2015.

[Gogula15] A. R. Gogulamudi, L. T. Clark, C. Farnsworth, S. Chellappa, and V. Vashishtha, “Architectural and Micro-architectural Techniques for Software Controlled Microprocessor Soft-error Mitigation,” 2015.

[Rama13] C. Ramamurthy, “Chip Level Implementation Techniques for Radiation Hardened Microprocessors,” Master’s thesis, Arizona State University, 2013.

[Koga93] R. Koga, S. D. Pinkerton, S. C. Moss, D. C. Mayer, S. Lalumondiere, S. J. Hansel, K. B. Crawford, and W. R. Crain, “Observation of single event upsets in analog microcircuits,” IEEE Trans. Nucl. Sci., vol. 40, no. 6, pp. 1838 – 1844, Dec. 1993.

[Ecof94] R. Ecoffet, S. Duzellier, P. Tastet, C. Aicardi, and M. Labrunee, “Observation of heavy ion induced transients in linear circuits,” Proc. IEEE NSREC Radiation Effects Data Workshop Record, pp. 72 – 77, 1994.

[Mavis02] Mavis, D.G.; Eaton, P.H.; , "Soft error rate mitigation techniques for modern microcircuits," Reliability Physics Symposium Proceedings, 2002. 40th Annual, vol., no., pp. 216- 225, 2002.

[Barn06] Barnaby, H. J., "Total-Ionizing-Dose Effects in Modern CMOS Technologies," Nuclear Science, IEEE Transactions on , vol.53, no.6, pp.3103-3121, Dec. 2006.

[Guert15] S. M. Guertin, M. Amrbar, and S. Vartanian, “Radiation Test Results for Common CubeSat Microcontrollers and Microprocessors,” *2015 IEEE Radiat. Eff. Data Work.*, vol. 91109, pp. 1–9, 2015.

[MIPS00] MIPS, “MIPS32 4Kc Processor Core Datasheet,” pp. 1–30, 2000.

[MIPS01] MIPS, “MIPS32 4K Processor Core Family Software User’s Manual,” 2001.

[Hind11] Hindman, N.D.; Clark, L.T.; Patterson, D.W.; Holbert, K.E.;, "Fully Automated, Testable Design of Fine-Grained Triple Mode Redundant Logic," Nuclear Science, IEEE Transactions on , vol.58, no.6, pp.3046-3052, Dec. 2011

[Hind09] Hindman, N.D.; Pettit, D.E.; Patterson, D.W.; Nielsen, K.E.; Xiaoyin Yao; Holbert, K.E.; Clark, L.T.; , "High speed redundant self-correcting circuits for radiation hardened by design logic," Radiation and Its Effects on Components and Systems (RADECS), 2009 European Conference on , vol., no., pp.465-472, 14-18 Sept. 2009.

[Sagg05] Saggese, G.P.; Wang, N.J.; Kalbarczyk, Z.T.; Patel, S.J.; Iyer, R.K.; , "An experimental study of soft errors in microprocessors," Micro, IEEE , vol.25, no.6, pp. 30-39, Nov.-Dec. 2005.

[Shin94] Shinichi Y., "A Radiation-Hardened 32-bit Microprocessor Based on the Commercial CMOS Process," IEEE Trans. Nucl. Sci., vol. 41, no. 6, pp. 2481-2486, 1894.

[Prit02] B. E. Pritchard, G. M. Swift, and A. H. Johnston, “Radiation effects predicted, observed, and compared for spacecraft systems,” Proc. IEEE NSREC Radiation Effects Data Workshop Record, pp. 7–17, 2002.

[Weav04] C. Weaver, J. Emer, S. Mukherjee, and S. Reinhardt, “Techniques to reduce the soft error rate of a high-performance microprocessor,” Proc. ISCA, 2004, pp. 264-27.

[Yao10]. X. Yao, L. Clark, D. Patterson, K. Holbert, “A 90 nm bulk CMOS radiation hardened by design cache memory,” IEEE Trans. Nucl. Sci, vol. 57, no. 4, pp. 2089-2097, Aug. 2010.

[Clark11] L. Clark, D. Patterson, N. Hindman, K. Holbert, and S. Guertin, “A dual mode redundant approach for microprocessor soft error hardness,” IEEE Trans. Nucl. Sci., vol. 58, no. 6, Dec. 2011, pp. 3018-3025.

[Vash15] V. Vashishtha, L. T. Clark, S. Chellappa, A. R. Gogulamudi, A. Gujja, and C. Farnsworth, “A Soft-Error Hardened Process Portable Embedded Microprocessor,” pp. 1–35, 2015.

[Rama15] C. Ramamurthy, S. Chellappa, V. Vashishtha, A. Gogulamudi, and L.T. Clark, “High Performance Low Power Pulse-Clocked TMR Circuits for Soft-Error Hardness,” vol. 62, no. 6, pp. 3040–3048, 2015.